ELSEVIER

Tutorial article

Comparison of encrypted control approaches and tutorial on dynamic systems using Learning With Errors-based homomorphic encryption

Junsoo Kim [a], Dongwoo Kim [b,*], Yongsoo Song [c], Hyungbo Shim [d], Henrik Sandberg [e], Karl H. Johansson [e]

[a] *Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul, Republic of Korea*
[b] *Western Digital Research, Milpitas, CA, USA*
[c] *Department of Computer Science and Engineering, Seoul National University, Seoul, Republic of Korea*
[d] *ASRI, Department of Electrical and Computer Engineering, Seoul National University, Seoul, Republic of Korea*
[e] *School of Electrical Engineering and Computer Science and Digital Futures, KTH Royal Institute of Technology, Stockholm, Sweden*

ARTICLE INFO

ABSTRACT

Encrypted control has been introduced to protect controller data by encryption at the stage of computation and communication, by performing the computation directly on encrypted data. In this article, we first review and categorize recent relevant studies on encrypted control. Approaches based on homomorphic encryption, multi-party computation, and secret sharing are introduced, compared, and then discussed with respect to computational complexity, communication load, enabled operations, security, and research directions. We proceed to discuss a current challenge in the application of homomorphic encryption to dynamic systems, where arithmetic operations other than integer addition and multiplication are limited. We also introduce a Learning With Errors based homomorphic cryptosystem called "Gentry-Sahai-Waters" scheme and discuss its benefits that allow for recursive multiplication of encrypted dynamic systems, without use of computationally expensive bootstrapping techniques.

1. Introduction

Networked control has enabled significant developments in numerous industrial fields, while it has also led to urgent issues related to cyber-security (Amin, Cárdenas, & Sastry, 2009; Sandberg, Amin, & Johansson, 2015). The more control systems have been connected to networks, the more possibilities for cyber-attacks have been discovered. Given the inherent cyber–physical nature of these systems, such attacks risk also the safe operation of the connected physical systems. Many incidents have been reported in recent years, including the StuxNet worm (Langner, 2011), false data injection to power grids (Liu, Ning, & Reiter, 2011), and a security breach in water sewage system (Slay & Miller, 2007).

Studies in security of control systems are distinguished from those in fault detection and isolation in the sense that the considered attacks are malicious, not random faults, and may be elaborately designed to avoid detection (Teixeira, Shames, Sandberg, & Johansson, 2015). A negligible probability of detection of a complex fault may be acceptable in safety studies, but in security such cases can be the point where the attacker deceives the detector. For example, zero-dynamics attacks (Pasqualetti, Dörfler, & Bullo, 2013) exploiting system models,

replay attacks (Mo & Sinopoli, 2009) exploiting signals, and covert attacks (Smith, 2015) exploiting both models and signals, have been reported. A common point in these attacks is that *the more the attacker learns of the target system, the more effective and theoretically undetectable attacks can be designed*. An extreme case is the covert attack presented in Smith (2015); if an adversary can compromise both the input and output communication using all the information of model and signals, then it can decouple the closed-loop of the plant and the controller, and manipulate the plant arbitrarily. This is while the controller (with any anomaly detector) cannot distinguish whether the received signals are compromised, or not (see Fig. 1).

From the motivation that data stored or being transmitted in networks may be used for advanced attack generation, the notion of *encrypted control* has been introduced (Farokhi, Shames, & Batterham, 2017; Kim et al., 2016; Kogiso & Fujita, 2015; Schulze Darup, Alexandru, Quevedo, & Pappas, 2021), which aims for security enhancement by protecting all data in networked controllers by encryption. Conventional encryption has been used in data transmission, as illustrated in Fig. 2(a), but computing devices in the network layer have been regarded as one of the most vulnerable parts. Since cryptosystems
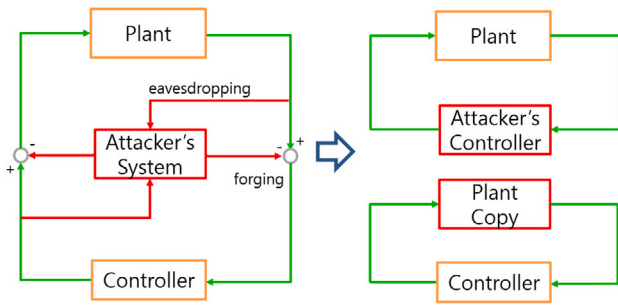

Fig. 1. A possible scenario (Smith, 2015) when all the information of system model and control signals is used for cyber-attacks.

allowing arithmetic operations was invented (Rivest, Adleman, Dertouzos, et al., 1978), more developed schemes (Cheon, Kim, Kim, & Song, 2017; ElGamal, 1985; Paillier, 1999) have been applied to control as in Farokhi et al. (2017), Kim et al. (2016) and Kogiso and Fujita (2015). The concept of "control operation over encrypted data", as illustrated in Fig. 2(b), is therefore a promising direction for improving cyber-security.

Benefits of the solution in Fig. 2(b) are that all data can be kept secure even when computation is performed, without decryption, and the secret key for decryption is not shared. Unauthorized parties, or possible network intruders, cannot learn any information through eavesdropping as long as they do not have the secret key, because encrypted signals cannot be distinguished from uniformly generated random signals. Furthermore, a typical type of cryptosystem used for encrypted operation, called homomorphic encryption, is known to be more beneficial than other security mechanisms. For example, compared to methods based on differential privacy (Dwork, 2008; Dwork, Roth, et al., 2014), it does not sacrifice precision of the encrypted messages, regardless of the increase of the security level. Also, many homomorphic encryption schemes are based on discrete lattice problems, such as Lyubashevsky, Peikert, and Regev (2010) and Regev (2009), and consider a stronger adversarial model than other schemes, such as chaotic cryptology techniques (Bourbakis & Alexopoulos, 1992; Matthews, 1989).

Nonetheless, the more operational abilities in recent cryptosystems are generally accompanied by increased costs in terms of computation or communication resource use. From an engineering point of view, application of cryptosystems to control thus should consider the following aspects for design.

- *Type of controller, required operations, and available resources.* Cryptosystems that allow arithmetic operations or guarantee a higher level of security usually require more data storage for encrypted data. The computational effort or communication overhead taken for each use of the arithmetic function may cost much more than usual operation over un-encrypted messages. Possibly, only one type of arithmetic function is supported, or only a limited number of function evaluations may be allowed. Thus, the design of encrypted control systems should consider the required type of operations and their complexity; the available amount of computation and communication resources; the time use for each arithmetic computation; whether the system is static or dynamic; the available amount of data storage; and if there are multiple units cooperating in the computation or there is only a single unit.
- *Security model (adversarial model).* The design should include an adversarial model, which determines from what attacks the system is protected. The adversary of interest can be an unauthorized party in the network, or an external hacker intruding the network. The adversary may aim for learning certain information by eavesdropping the communication lines or networked devices; or for

manipulating or forging a portion of data with a certain objective in mind. Depending on the number of units and the structure of the design, the security objective can be to protect the data from external parties only, or to conceal it from participating units who may be curious about other participants' private information.

Considering the above aspects, the designer of an encrypted control system chooses an appropriate cryptosystem, which may be computationally fast with only limited arithmetic support, may have much more operation abilities with requirement of large amounts of resources, or can be suitable for centralized or distributed computation schemes, respectively. Usually, there are trade-offs between the cryptosystems, in terms of use of resources, level of security, or enabled encrypted operations. Relevant research efforts try to overcome this trade-off; by re-constructing a computation system which exploits only a limited type of operation, by developing a novel cryptosystem that is more appropriate for a certain application, or by proposing a protocol that guarantees an improved level of security. In short, the efforts try to improve the security or computational efficiency, while not losing operational abilities.

Contribution and organization: This article provides both a general review of the field of encrypted control and a comprehensive introduction of the specific topic of dynamic system design based on homomorphic encryption. In Section 2, we discuss main approaches for encrypted control, which are based on *homomorphic encryption, multi-party computation, or secret sharing*, respectively. These cryptosystems, as tools for encrypted control, are briefly introduced, and the approaches and their application to control are compared. Trade-offs between computation speed, operation ability, and security are also discussed. In Section 3, we focus on the homomorphic encryption based approach and introduce recent methods for linear dynamic systems. A tutorial on a Learning With Errors (LWE) based homomorphic encryption scheme is first provided, and we discuss its benefit in terms of recursive multiplication. Then, we review a method that implements linear systems by exploiting only the addition and multiplication abilities of cryptosystem. Finally, Section 4 concludes the paper.

Notation: Let $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{R}$ denote the set of natural numbers, integers, and real numbers, respectively. The (component-wise) floor and round functions are denoted by $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$, respectively. For $m \in \mathbb{N}$ and $n \in \mathbb{N}$, we let $0_{m \times n} \in \mathbb{R}^{m \times n}$ be the zero matrix, and $I_n \in \mathbb{R}^{n \times n}$ be the identity matrix. The set of integers modulo $q \in \mathbb{N}$ is denoted by $\mathbb{Z}_q$, and the (component-wise) modulo operation is defined by $v \bmod q := v - \lfloor v/q \rfloor q$ for $v \in \mathbb{Z}^m$. We make use of "biased" modulo operation defined as

$$v \quad \bmod (q, v_0) := v - \left\lfloor \frac{v - v_0}{q} \right\rfloor q \tag{1}$$

for $v \in \mathbb{Z}^m$ and $v_0 \in \mathbb{R}^m$, so that each component of the outcome is greater than or equal to that of v_0, and less than that of $v_0 + q$. We define $\mathrm{col}\{v_i\}_{i=1}^n = [v_1^\top, \ldots, v_n^\top]^\top$ for column vectors $\{v_i\}_{i=1}^n$ (or scalars), and the (induced) infinity norm of a vector or a matrix is denoted as $\| \cdot \|$.

2. Encrypted control approaches

This section categorizes the relevant works into four categories according to the underlying cryptographic primitives; homomorphic encryption, fully homomorphic encryption, secret sharing, and multi-party computation, together with a brief introduction to each primitive and approach. Then, we provide discussions and comparisons between them.
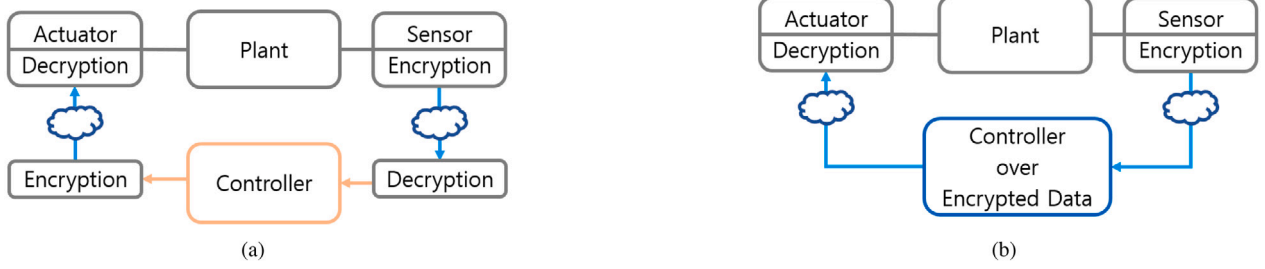
Fig. 2. (a) Data protection by conventional encryption. (b) Configuration of encrypted control.

2.1. Homomorphic encryption based control

We begin with a brief introduction on homomorphic encryption. Let a cryptosystem be denoted by $(\mathbb{Z}_q, C, \mathsf{Enc}, \mathsf{Dec})$, where the set $\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$, $q \in \mathbb{N}$, is the plaintext (un-encrypted message) space, C is the ciphertext (encrypted) space, $\mathsf{Enc} : \mathbb{Z}_q \to C$ and $\mathsf{Dec} : C \to \mathbb{Z}_q$ are the encryption and decryption algorithms, respectively. We omit the argument of encryption and decryption keys, for simplicity.

Homomorphic properties of cryptosystems imply that the encryption and decryption algorithms are "homomorphisms", which preserve algebraic structure with respect to a certain arithmetic function. Let us assume that the cryptosystem $(\mathbb{Z}_q, C, \mathsf{Enc}, \mathsf{Dec})$ is additively homomorphic, which means that the algorithms Enc and Dec are homomorphic with respect to the addition operation; there exists a binary function $* : C \times C \to C$ over ciphertexts such that

$$\mathsf{Dec}(\mathbf{c}_1 * \mathbf{c}_2) = \mathsf{Dec}(\mathbf{c}_1) + \mathsf{Dec}(\mathbf{c}_2) \mod q, \qquad \forall \mathbf{c}_1 \in C, \ \forall \mathbf{c}_2 \in C,$$

which implies that

$$\mathbf{c} = \mathsf{Enc}(x_1) * \mathsf{Enc}(x_2) \implies \mathsf{Dec}(\mathbf{c}) = x_1 + x_2 \mod q,$$
$$\forall x_1 \in \mathbb{Z}_q, \ \forall x_2 \in \mathbb{Z}_q. \quad (2)$$

It means that whenever we decrypt the outcome of the operation $*$ over ciphertexts, we obtain the same addition outcome over plaintexts, so it enables to perform the addition directly over encrypted messages, without decryption.

The ability of addition is a basic property, but by exploiting it, multiplication by (un-encrypted) constant numbers can also be performed. Given a natural number $k \in \mathbb{N}$, we define

$$k \cdot \mathbf{c} := \overbrace{\mathbf{c} * \cdots * \mathbf{c}}^{k \text{ times}}.$$

And, we extend the definition for an integer $k \in \mathbb{Z}$, by

$$k \cdot \mathbf{c} := (k \mod (q, 1)) \cdot \mathbf{c},$$

where the operation $\mathrm{mod}(q, 1)$ defined from (1) ensures that $k \mod (q, 1) \geq 1$. Then, it follows that

$$\mathsf{Dec}(k \cdot \mathbf{c}) = (k \mod (q, 1)) \cdot \mathsf{Dec}(\mathbf{c}) \mod q$$
$$= k \cdot \mathsf{Dec}(\mathbf{c}) \mod q$$

holds for all $k \in \mathbb{Z}$ and $\mathbf{c} \in C$, because $k \mod(q, 1) = k + dq$ with some $d \in \mathbb{Z}$.

It enables multiplication by integer matrices as well; for an integer matrix $K = [k_{ij}] \in \mathbb{Z}^{m \times n}$ and a set $\mathbf{x} = \{\mathsf{Enc}(x_i)\}_{i=1}^{n}$ of encrypted messages of $x_i \in \mathbb{Z}_q$, $i = 1, \ldots, n$, we define

$$K \cdot \mathbf{x} = \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1n} \\ k_{21} & k_{22} & \cdots & k_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ k_{m1} & k_{m2} & \cdots & k_{mn} \end{bmatrix} \cdot \begin{bmatrix} \mathsf{Enc}(x_1) \\ \mathsf{Enc}(x_2) \\ \vdots \\ \mathsf{Enc}(x_n) \end{bmatrix}$$
$$:= \begin{bmatrix} (k_{11} \cdot \mathsf{Enc}(x_1)) * (k_{12} \cdot \mathsf{Enc}(x_2)) * \cdots * (k_{1n} \cdot \mathsf{Enc}(x_n)) \\ (k_{21} \cdot \mathsf{Enc}(x_1)) * (k_{22} \cdot \mathsf{Enc}(x_2)) * \cdots * (k_{2n} \cdot \mathsf{Enc}(x_n)) \\ \vdots \\ (k_{m1} \cdot \mathsf{Enc}(x_1)) * (k_{m2} \cdot \mathsf{Enc}(x_2)) * \cdots * (k_{mn} \cdot \mathsf{Enc}(x_n)) \end{bmatrix}, \quad (3)$$

which is simply a component-wise constant multiplication and consecutive additions over encrypted data. Then, it can also be easily verified that $\mathsf{Dec}(K \cdot \mathbf{x}) = K \cdot [x_1, x_2, \ldots, x_n]^\top \mod q$, for all $K \in \mathbb{Z}^{m \times n}$ and $x \in \mathbb{Z}_q^n$.

Thanks to the property of allowing computation over ciphertexts without the secret key nor decryption, the computation can be assigned to an operation unit who is honest to do the computation correctly while being curious about the transmitted data, or to a networked unit which can be possibly accessed by unauthorized third parties. In addition to the described homomorphic properties, there are also encryption schemes (Boneh, Goh, & Nissim, 2005; Brakerski, Gentry, & Vaikuntanathan, 2014; Brakerski & Vaikuntanathan, 2011; Cheon et al., 2017; Dijk, Gentry, Halevi, & Vaikuntanathan, 2010; Fan & Vercauteren, 2012) which enable to compute both the addition and a limited number of multiplication on encrypted data. Those schemes are sometimes called somewhat or leveled homomorphic encryption to be distinguished from (partially) homomorphic encryptions allowing addition only, but we denote both of them, simply, as homomorphic encryption.

The application of homomorphic encryption for control has been introduced in Farokhi et al. (2017), Kim et al. (2016) and Kogiso and Fujita (2015). Since homomorphic cryptosystems support addition and/or multiplication, most controllers based on homomorphic encryption considers linear operation (matrix multiplication) or polynomial functions (represented with small number of additions and multiplications) only; see Fig. 3(a) for an example case of linear controllers based on homomorphic encryption. It has also been used for implementing model predictive control (Schulze Darup, Redder, Shames, Farokhi, & Quevedo, 2017), data-driven control (Alexandru, Tsiamis, & Pappas, 2020), and reinforcement learning based control (Suh & Tanaka, 2021), where the computation circuits are represented with linear functions or low degree polynomials.

Operation unit and attack model: The model of the controller is usually considered as a single operation unit who performs the computation over encrypted data without accessing the secret key. It is often specified as a networked controller under possible eavesdropping attack (which tries to learn private control data) or considered as a cloud-based controller performing the computation on behalf of the controller designer, which should not learn any private information through the computation. In short, adversaries of homomorphic encryption based controllers can be both external eavesdroppers and the controller itself, and cryptosystem parameters are determined for a desirable level of security.

Challenges: A main issue is that encrypted implementation may be limited to only addition or multiplication over integers. Even for homomorphic encryption allowing both addition and multiplication, many of them allow evaluating a limited number of multiplications only. As a consequence, implementation of dynamic systems that iteratively compute and update the state has been considered as a challenge, and representing diverse nonlinear systems using only addition and multiplication is currently an open problem.

Even for linear systems, dynamic systems implemented over encrypted data may be incapable of operating for an infinite time horizon. This is because, based on homomorphic encryption only, multiplication
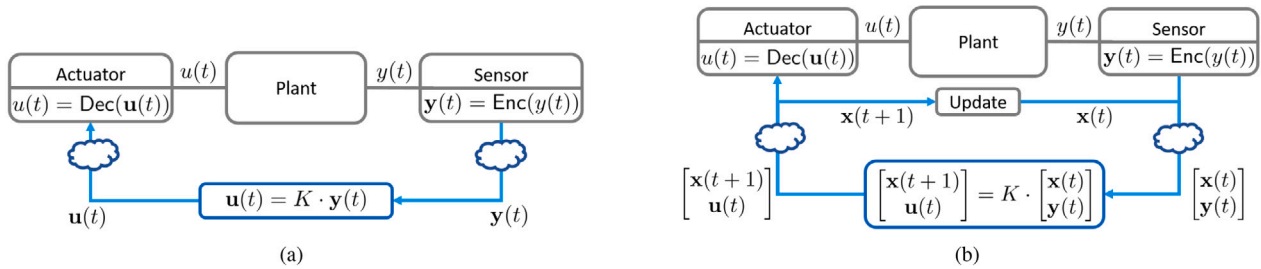
Fig. 3. (a) Homomorphic encryption based controllers for static feedback. (b) Dynamic encrypted controller implementation using additional transmission lines for the state.
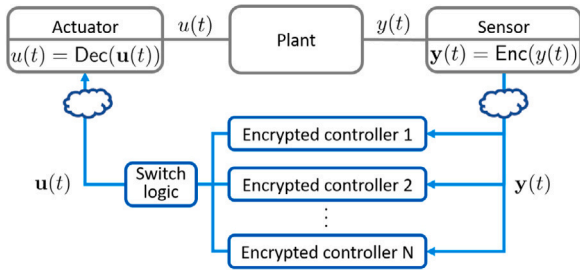
Fig. 4. Multiple encrypted linear controllers with switching function.

by non-integer numbers can be performed only a finite number of times, so that the dynamic operation that multiplies the state by non-integer numbers cannot be continued for an infinite time horizon (see the example described in (32)). Fig. 3(b) describes a method for dynamic controllers considered in initial studies, which uses additional transmission lines for the state; assuming that the encrypted state of the controller can be transmitted to the device having the decryption key, the state can be decrypted and re-encrypted and transmitted back to the controller. However, the limitation is that the system cannot continue the operation without the presence of the decryption key, and the re-encryption requires additional use of communication lines which may be proportional to the dimension of the signal. Further discussions and related methods can be found in Section 3.2.

2.2. Fully homomorphic encryption based methods

As introduced, homomorphic encryption supporting a finite number of operations has limitations on its applicability. The first fully homomorphic encryption, with which one can perform arbitrary computation without limitation, has been presented in Gentry (2009). Then, the main concern has been reducing the computational cost, and follow-up works have been proposed toward their practical use (Brakerski et al., 2014; Cheon et al., 2017; Chillotti, Gama, Georgieva, & Izabachene, 2016; Ducas & Micciancio, 2015; Fan & Vercauteren, 2012), with several library implementations. The crucial improvement from the previous homomorphic encryption (with limitations) is the introduction of "bootstrapping" procedure[1] (Ducas & Micciancio, 2015; Gentry, Halevi, & Smart, 2012) which refreshes (without a secret key) a ciphertext into a new one with which one can continue the computation. Many encryption schemes referred to in the previous section can be fully homomorphic, by adding the bootstrapping procedure.

Thanks to the bootstrapping, any operation circuits in digital computers can be implemented and run over encrypted data for an infinite time horizon, since any logical functions with Boolean variables can be represented using addition and multiplication. Nonetheless, the

---

[1] Conceptually, it can be understood as an encrypted evaluation of the decryption operation on the ciphertext, resulting in a new ciphertext having the same message.

computational complexity of bootstrapping has been a critical issue, and its requirement on the computation resource currently hinders it from being used in practice.

The bootstrapping was first used for linear dynamic systems in Kim et al. (2016). The latency time for bootstrapping may be larger than the sampling period, but as described in Fig. 4, multiple identical controllers are designed and the time for bootstrapping is scheduled in a different way, so that the same output can be generated all the time even when some of them are performing bootstrapping. Many homomorphic encryption based works (Alexandru et al., 2020; Fritz, Fauser, & Zhang, 2019; Suh & Tanaka, 2021; Zhou et al., 2020) mentioned in the previous section also claim that introduction of bootstrapping enables the encrypted systems to update the state without interacting with an entity holding the secret key.

Operation unit and attack model: Since fully homomorphic encryption is simply a homomorphic encryption with further evaluation capabilities, the operation model considered is the same as the case of homomorphic encryption without use of bootstrapping. It also considers a single operation unit who should not learn any information from the computation, while all the control data are kept encrypted so that it is protected from external parties as well. A difference in practice is that it should be able to use a large amount of computational or memory resources, for bootstrapping.

Challenges: Main challenge of fully homomorphic encryptions is to reduce the computational cost for bootstrapping. A couple of issues regarding control operation can be listed; first, operations consisting of addition, multiplication, and bootstrapping, may not be efficient for non-polynomial functions, such as, comparison, if-else conditionals, or transcendental functions, for which lots of arithmetic operations may be required for the representation and result in impractical evaluation cost. A countermeasure would be to represent the given operation circuit to an "arithmetic-friendly form", so that the evaluation cost can be reduced. And, since bootstrapping constitutes a major cost in fully homomorphic encryption based schemes while it has been essentially used, a momentous challenge will be to reduce the number of uses of bootstrapping.

2.3. Secret sharing based schemes

Now, we describe another computation protocol based on secret sharing, where private data are distributed in the form of "shares" to two or more parties so that an individual party cannot get any information about the data. Still, each party can perform appropriate computation on its share and generate the outcome from which the owner of the data can reconstruct the computation outcome; in other words, there is a homomorphic property on the shares, so that the computation of control signals can be offloaded to outsourced operation units, without disclosing any information on the input and output.

The most basic secret sharing with two computing parties, supporting additions, is described as follows. Let $m \in \mathbb{Z}_q$ be a message that should be kept securely. A share for the first party is generated as $c_1 := m + r \bmod q$, where $r \in \mathbb{Z}_q$ is a random number sampled from the uniform distribution over the set $\mathbb{Z}_q$. And, the share for the other
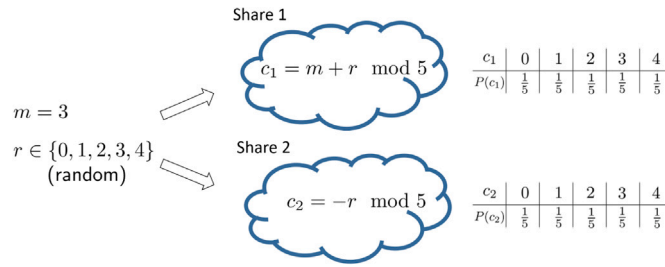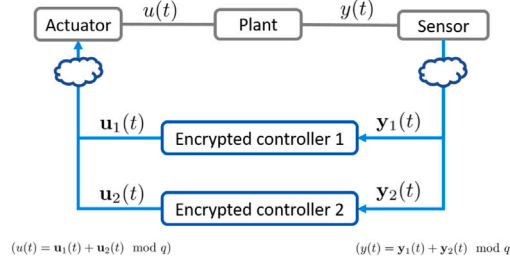
Fig. 5. A basic example of secret sharing.

Fig. 6. Configuration of encrypted controller based on secret sharing with two operation units.

party is given by $c_2 := -r \bmod q$, so that $m = c_1 + c_2 \bmod q$. By doing so, the distribution of each share on each party follows the uniform distribution over the set $\mathbb{Z}_q$, which means that referring to only one share does not give any information about the message m. We note that a new random number must be sampled whenever a message is split to shares. See Fig. 5 for an example with $\mathbb{Z}_q = \mathbb{Z}_5 = \{0, 1, 2, 3, 4\}$ and $m = 3$.

The method itself is quite simple, but this scheme as a cryptosystem is homomorphic with respect to addition and integer multiplication; let two messages $m \in \mathbb{Z}_q$ and $m' \in \mathbb{Z}_q$ be split into two shares, as $m = c_1 + c_2 \bmod q$ and $m' = c_1' + c_2' \bmod q$, respectively. And, let the first shares $c_1 \in \mathbb{Z}_q$ and $c_1' \in \mathbb{Z}_q$ be kept by the first operation unit and let the other shares $c_1' \in \mathbb{Z}_q$ and $c_2' \in \mathbb{Z}_q$ be kept by the second operation unit. Then, the addition operation ($m + m' \bmod q$) can be performed by the operation units in parallel, as

Operation unit 1 : $\quad c_1^{\mathrm{add}} = c_1 + c_1' \quad \bmod q$

Operation unit 2 : $\quad c_2^{\mathrm{add}} = c_2 + c_2' \quad \bmod q$

where the operation units cannot learn any information of m, m', or $m + m' \bmod q$. When the outcomes c_1^{add} and c_2^{add} as shares are collected from both the units, then the plaintext outcome can be restored, because $(m + m' \bmod q) = (c_1^{\mathrm{add}} + c_2^{\mathrm{add}} \bmod q)$. And, for a vector $\vec{m} \in \mathbb{Z}_q^n$ of n-messages split as $\vec{m} = \vec{c}_1 + \vec{c}_2 \bmod q$ by generating n-random numbers in $\mathbb{Z}_q$, multiplication by (un-encrypted) integer matrices can also be done by the operation units in parallel; for a matrix $K \in \mathbb{Z}^{m \times n}$, let the operation units compute

Operation unit 1 : $\quad c_1^{\mathrm{mult}} = K \cdot \vec{c}_1 \quad \bmod q$

Operation unit 2 : $\quad c_2^{\mathrm{mult}} = K \cdot \vec{c}_2 \quad \bmod q.$

Then, it is easy to check that $(K \cdot \vec{m} \bmod q) = (c_1^{\mathrm{mult}} + c_2^{\mathrm{mult}} \bmod q)$. For general schemes, one may refer to Shamir's secret sharing (Shamir, 1979), and many extended results and methods can be found in literature (Beimel, 2011; Brickell, 1989; Karnin, Greene, & Hellman, 1983).

We have seen that secret sharing enables offloading linear operations without disclosing information, given that the computing parties do not collude with others. Contrary to homomorphic encryptions demanding a relatively large amount of storage and computational resources on the operation device, secret sharing based methods can be a good alternative for applications where computational resources are limited but multiple operation units are available. For example,

it is proposed in Schulze Darup and Jager (2019) that the use of secret sharing for encrypted control can reduce the time consumption significantly, per each unit of operation (compared to homomorphic encryption). Fig. 6 describes a linear controller based on secret sharing; the sensor measurement $y(t)$ is distributed, as shares $y_1(t)$ and $y_2(t)$, to two controllers, and each controller performs the linear operation and sends the outcome to the actuator, respectively.

Secret sharing has also been employed for dealing with privacy problems in distributed computation protocols, where multiple participants collaboratively evaluate a common function of interest. For example, average consensus can be considered as in Fig. 7(a), where the participants compute the average of the individual values by communicating with their neighbors. In terms of privacy, the goal is to perform the computation while each individual value is not disclosed to other parties. Methods based on differential privacy have been frequently employed as in Mo and Murray (2016), Nozari, Tallapragada, and Cortés (2017), where a trade-off between privacy and performance is inevitable, depending on the size of injected noises. Then, the secret sharing has been introduced for distributed systems; for example, Fig. 7(b) describes a method proposed in Wang (2019) for privacy-preserving average consensus. By splitting each individual value to two shares, transmitting only one of them to the neighbors directly, and reflecting the other share to the protocol "indirectly", the distributed protocol can yield the same outcome without sacrificing the privacy of each individual value. An advantage compared to differential privacy is that there is no performance penalty for keeping the data private.

Operation unit and attack model: Secret sharing based control systems introduce multiple operation units which perform the computation on the distributed shares in parallel, and thus they consider an adversarial model compromising some portion of the units. Specifically, "(t, N)-threshold scheme" denotes that an adversary needs to get t-shares out of the N-shares (distributed to the units) to recover the message. For instance, the example described in Fig. 5 can be seen as (2,2)-threshold secret sharing.

Challenges: Whereas secret sharing allows for efficient computation between the shares of messages, the issue of possible collusion between the units that reveals the value of the message is a main issue in this framework. Furthermore, additional potential risk in practice would be that, if there is an external party (not participating in the computation) who succeeds in collecting the operation units' information, then the private data can also be exposed. Another challenge is that, analogous
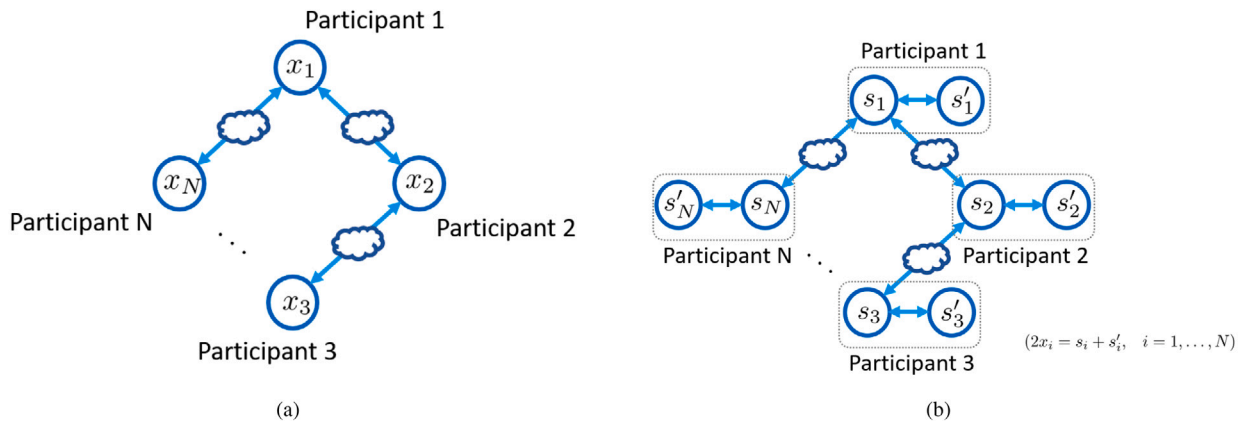
Fig. 7. (a) Conventional distributed protocol. (b) Distributed protocol based on secret sharing.
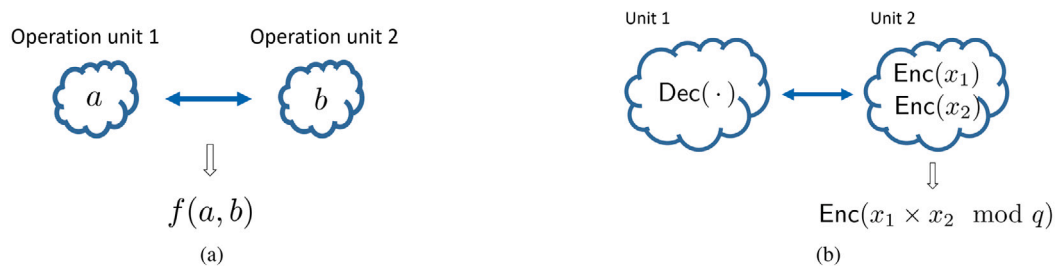

Fig. 8. (a) Configuration of two-party computation. (b) Encrypted multiplication based on two-party computation.

to methods solely based on homomorphic encryption, general nonlinear functions over secret shares may be hard to implement and currently they are limited. There is a trade-off between further operation abilities on the shares and sacrifice of the security, so that it can be studied and improved in the future; to perform operations other than addition, interactions between the computing units or relaxation of the security model is required. For example, in Schlor, Hertneck, Wildhagen, and Allgöwer (2021), computation of polynomials of degree N results in a $(2, N)$-threshold secret sharing, where the private message is revealed if any two of the shares are collected.

There are several works exploiting secret sharing to enhance the privacy of control systems, where many of them also employ homomorphic encryption or other cryptographic schemes, together with use of communication between the operation units. We refer to them as "multi-party computation" based schemes, which will be discussed in the next subsection. In general, secret sharing is often regarded as a sort of multi-party computation scheme, as it also utilizes multiple parties for the computation. Nonetheless, for the sake of detailed comparison and discussion in Section 2.5, we distinguish the secret sharing algorithms from the multi-party computation; secret sharing desirably does not utilize communication between the parties during the computation, whereas for the multi-party computation, the parties essentially communicate with other parties and collaborate for each unit of computation.

2.4. Multi-party computation based control

Finally, we introduce approaches based on multi-party computation. The terminology multi-party computation is usually used broadly and often includes the homomorphic encryption and secret sharing based methods discussed in the previous subsections. Nonetheless, by "multi-party computation", throughout the article, we mean a narrower framework employing homomorphic encryption, secret sharing, and/or other cryptographic primitives, where the parties cooperate for each unit of computation, by several interactions and communications between them.

The scheme is to let multiple participants collaboratively evaluate a function, where the inputs are given from the participants, but the privacy of each input should be kept. That is, each participant's input should be used for the joint computation, but it should not be learned by any other participant.[2] See Fig. 8(a) describing a configuration of two-party computation as a basic case; the two units jointly compute the value of $f(a, b)$ while the arguments a and b are not informed by the other unit.

Fig. 8(b) shows a basic example of encrypted multiplication by two-party computation. Let Unit 2 have two encrypted numbers, Unit 1 have the decryption key, and let the encryption Enc be additively homomorphic but not multiplicatively homomorphic. That is, Unit 2 can perform encrypted addition using the function $*$ as in (2), but it cannot do the multiplication for the two encrypted messages $\mathsf{Enc}(x_1)$ and $\mathsf{Enc}(x_2)$ directly, by itself. Then, the objective is to let Unit 2 obtain the multiplication outcome of two messages as encrypted, with help of Unit 1, while Unit 1 should not learn the values of x_1 and x_2, and Unit 2 should not obtain Unit 1's decryption key.

Then, Fig. 9 shows how the two-party computation for encrypted multiplication is performed. Essentially, Unit 2 will send the two messages to Unit 1, and Unit 1 will do the multiplication after decrypting the received ciphertexts. Unit 1 should not know the message values, so Unit 2 generates two random numbers r_1 and r_2 and add to $\mathbf{x}_1$ and $\mathbf{x}_2$ using the additively homomorphic property, and then transmit the "masked" messages. Then, even though Unit 1 decrypts the messages, it cannot learn anything from the values $(x_1 + r_1 \bmod q)$ and $(x_2 + r_2 \bmod q)$. Nevertheless, now Unit 1 can multiply the decrypted messages, encrypt the outcome again, and send it back to Unit 2. Finally, the transmitted ciphertext $\mathbf{z}$ contains the message as $(x_1 + r_1)(x_2 + r_2) \bmod q$, and since Unit 2 has the information of r_1, r_2, $\mathsf{Enc}(x_1)$,

---

[2] The goal can be defined formally with a simulation paradigm which comprises broader security guarantees such as detecting malicious behaviors with data privacy (we refer to Lindell (2017)). In this article, we rather focus on the data privacy aspects.
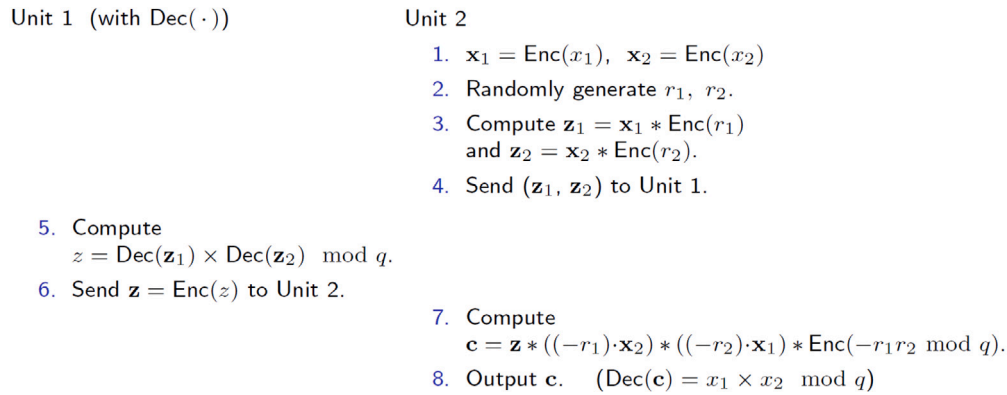
Unit 1 (with Dec($\cdot$))

Unit 2

1. $\mathbf{x}_1 = \mathsf{Enc}(x_1)$, $\mathbf{x}_2 = \mathsf{Enc}(x_2)$

2. Randomly generate r_1, r_2.

3. Compute $\mathbf{z}_1 = \mathbf{x}_1 * \mathsf{Enc}(r_1)$
   and $\mathbf{z}_2 = \mathbf{x}_2 * \mathsf{Enc}(r_2)$.

4. Send $(\mathbf{z}_1, \mathbf{z}_2)$ to Unit 1.

5. Compute
   $z = \mathsf{Dec}(\mathbf{z}_1) \times \mathsf{Dec}(\mathbf{z}_2) \mod q$.

6. Send $\mathbf{z} = \mathsf{Enc}(z)$ to Unit 2.

7. Compute
   $\mathbf{c} = \mathbf{z} * ((-r_1) \cdot \mathbf{x}_2) * ((-r_2) \cdot \mathbf{x}_1) * \mathsf{Enc}(-r_1 r_2 \mod q)$.

8. Output $\mathbf{c}$. $(\mathsf{Dec}(\mathbf{c}) = x_1 \times x_2 \mod q)$

Fig. 9. A basic method for two-party encrypted multiplication.

and $\mathsf{Enc}(x_2)$, it can cancel out the terms $r_1 x_2$, $r_2 x_1$, and $r_1 r_2$, over encrypted data. As a consequence, Unit 2 obtains the outcome whose decryption is equal to $x_1 \times x_2 \mod q$. It is also clear that Unit 2 does not learn any additional information because it only computes over encrypted messages during the process.

We have seen that two-party computation enables encrypted multiplication, despite that the ability of multiplication is not given from the homomorphic cryptosystem. In general, a main advantage of multi-party computation is that *it can allow various functions other than addition and multiplication while exploiting additively homomorphic encryption only, by use of multiple parties and communication between them*. Many results have contributed for enabling various functions (Bogetoft et al., 2009; Cramer, Damgård, et al., 2015; Dahl, Ning, & Toft, 2012; Damgård, Fitzi, Kiltz, Nielsen, & Toft, 2006; Nishide & Ohta, 2007), such as division by integers, inversion of encrypted matrices, and comparison or maximum operations. It is also notable that usually multi-party computation does not make use of bootstrapping of fully homomorphic encryption so that the required amount of computational resources is not exhaustively large, but it rather exploits communication resources several times for each unit of arithmetic function.

Meanwhile, similarly to the framework of secret sharing, a crucial assumption on multi-party computation is that the parties do not collude with each other. For example, supposing that Units 1 and 2 in the algorithm in Fig. 9 are "outsourced computers" performing the encrypted computation on client's private data x_1 and x_2, the information will be exposed once the units collude and Unit 1 decrypts Unit 2's encrypted data.

A basic configuration of the two-party computation based control scheme is described in Fig. 10. It considers two collaborative operation units consisting of an "encrypted controller" who stores encrypted control data and performs arithmetic supported from the employed homomorphic encryption, and a "computation assistant" who helps advanced operations that cannot be done with the homomorphic property of cryptosystem solely, by use of two-party computation techniques. In case the computation assistant is supposed to have the decryption key of the cryptosystem, the actuator can take the role, assuming that the communication between the actuator and the encrypted controller can be bi-directional (for this case, the structure of system becomes similar with that of Fig. 3(b), where additional communication is used for "re-encrypting" the state).

Multi-party computation based control employs multiple operation units, which are often regarded as "out-sourced" computers, who performs the computation on behalf of the controller designer. Thanks to its utility enabling further advanced arithmetic functions, it has been applied to nonlinear controllers which need more than addition and multiplication for operating. For example, it has been used for encrypted implementations of "implicit" model predictive controllers where projection operation is needed for constrained optimizations (Alexandru, Morari, & Pappas, 2018; Alexandru & Pappas,

2020), or extended Kalman filters where matrix inversions and comparison operations are required (Gonzalez-Serrano, Amor-Martın, & Casamayon-Anton, 2014).

Attack model: Since more than one parties are involved, the adversarial model for multi-party computation framework is similar to that of secret sharing. It can be described similarly according to the number of parties that the adversary is required to compromise to recover the underlying messages. Usually, the security model also considers the behavior that the adversary can carry on; "honest-but-curious" model assumes that the adversary follows the protocol correctly (but tries to infer useful information from the observation) whereas "malicious" model assumes that the adversary may not follow the protocol correctly and perform an arbitrary behavior (for this case, the protocol aims to enable other participants to detect such behaviors).

Challenges: Compared to fully homomorphic encryption (which achieves a similar functionality), multi-party computation methods in general require much less computational resources but exploit more communication costs for each unit of arithmetic functions. Interestingly, similarly to (fully) homomorphic encryption, evaluating non-arithmetic functions such as division, transcendental functions, and conditional expressions may require much more communications (and computations) than arithmetic functions. While there are many specialized multi-party protocols for evaluating a unit of function, it often requires combining them together to design an encrypted system. We also remark that constructing a system based on multi-party computation protocol usually involves more complex security analysis and proofs, even in the same honest-but curious model, than that based on (fully) homomorphic encryption, due to the presence of multiple parties and interactions between them where the adversary tries to compromise.

2.5. Discussion and comparison

Finally, Table 1 puts all the discussed approaches and their features together, and compares them with each other, where general trade-offs between the encrypted control approaches are found. To this end, for simplicity, let us abbreviate homomorphic encryption as "HE", fully homomorphic encryption as "FHE", and secret sharing as "SS", in this subsection. Multi-party computation protocols, utilizing homomorphic encryption, are particularly denoted by "MPC-HE", so that they are distinguished from secret sharing.

One natural observation is that "the more resources are utilized, then the more operation abilities (over encrypted data) the scheme obtains". Comparing HE and FHE, it can be seen that FHE utilizes much more computational resources, but it guarantees that any sort of operation circuit can be implemented. Comparing HE and MPC-HE, on the other hand, it can be understood that additional use of communication resources for MPC-HE schemes mitigates the limited operation issue of HE; e.g., the ability of HE is limited to addition and
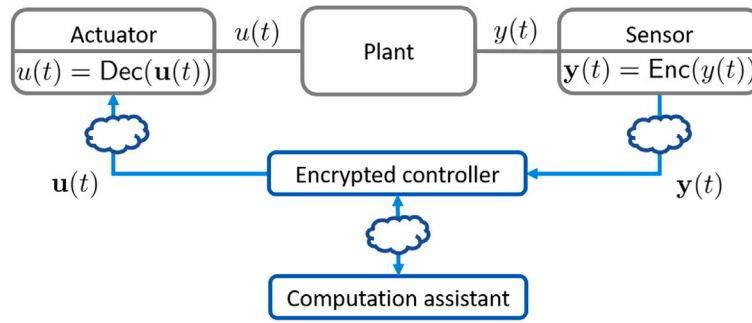
Fig. 10. Configuration of encrypted controller based on two-party computation.

Table 1
Trade-offs between cryptographic tools (compared to HE).

|	HE	FHE	SS	MPC-HE
Feature	–	Computation-intensive	Operation without homomorphic encryption	Communication-intensive
Operation unit	Single	Single	Multiple	Multiple
Strengths	–	No limitation on operations	Computationally efficient	Advanced operations without bootstrapping
Weaknesses	Limitation on operations	High computational complexity	Limitation on operations, collusion issues	Communication delay, collusion issues
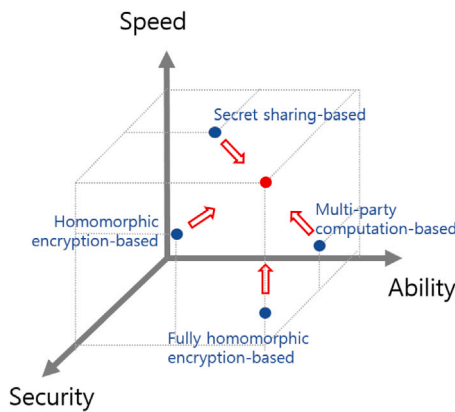
Fig. 11. Comparison of encrypted control approaches and future research directions.

multiplication over integers, but MPC-HE can implement many more advanced functions.

Another aspect in comparison is that "use of multiple operation units" brings improvement of computational efficiency, but it also brings possible security issues together. Comparing HE and MPC-HE again, MPC-HE takes advantage of multiple parties so that it obtains more operation abilities, and comparing HE and SS, SS also introduces multiple operation units so that it improves computational efficiency in the sense that it keeps the ability of linear operation still amenable, despite that it does not use homomorphic encryption. However, a main opportunity cost for utilizing multiple operation units is possible security issues; it should be guaranteed that the participating units never collude with the others, and it might also be vulnerable to eavesdropping from external adversaries.

A similar observation can be made in a different way. Let Fig. 11 compare the four approaches, in terms of speed (operation amount per unit of time), ability (enabled operations over encrypted data), and security (mainly considers issues of collusion and external eavesdropping). In terms of "speed", SS may be a good option if the system can be implemented using multiple computers. Regarding "ability", if the encrypted controller design includes complicated nonlinear functions,

either FHE based method or MPC-HE based protocol can be considered, with intensive use of computational or communication resources, respectively. And, for applications in which use of multiple operation units is not allowed because of security matters, use of HE or FHE would be appropriate. In short, each encrypted control application should consider its required sort of operations, available resources, and appropriate security model, for choosing its approach.

Lastly, Fig. 11 also considers future research directions for the approaches, which would be to deal with and improve on their respective weaknesses. HE based schemes would aim for overcoming the constraint of limited operation, and proposing practical bootstrapping techniques would be of main interest for FHE based methods. For the sake of control application, SS and MPC-HE based schemes should resolve possible security issues due to collusion or external eavesdropping. At the same time, developing nonlinear functions using SS and reducing the communication overhead with MPC-HE would be of challenges, respectively.

3. Introduction to linear dynamic systems using Learning With Errors-based homomorphic encryption

So far we have discussed broad approaches on encrypted control and current challenges. To handle those issues, new problems that have not been considered in the control community are often formulated. In this section, let us be more focused on the approach based on homomorphic encryption only, and introduce a specific problem and current solutions, on implementation of dynamic systems over encrypted data.

Implementing an encrypted controller is to convert a "given" controller over plaintext (described in Fig. 2(a)) to a system over ciphertext (as Fig. 2(b)). Let a linear dynamic controller has been designed and given as

$$x(t + 1) = Fx(t) + Gy(t)$$
$$u(t) = Hx(t) + Jy(t) \quad (4)$$
$$x(0) = x_0$$

where $x(t) \in \mathbb{R}^{\ell}$ is the state, $y(t) \in \mathbb{R}^{p}$ is the sensor measurement transmitted from the plant (input of the controller), $u(t) \in \mathbb{R}^{m}$ is the actuation input fed back to the plant (output of the controller), and the matrices $\{F, G, H, J\}$ and the initial state $x_0 \in \mathbb{R}^{\ell}$ are given

as controller parameters. Throughout the paper, we assume that the trajectories of the signals $\{x(t), y(t), u(t)\}_{t=0}^{\infty}$ are bounded.

Then, the implementation of the controller (4) over encrypted data can be divided into two steps.

1. *Conversion of given controller to operate over integers.* As discussed in Section 2.1, most homomorphic cryptosystems allow no more than addition and multiplication over encrypted integers, unless bootstrapping (of fully homomorphic encryption) is used. Thus, the given system (4) with the parameters $\{F, G, H, J, x_0\}$, which are generally non-integers, should be converted to operate over integers using addition and multiplication only. The input–output relation (performance) of the re-constructed controller should be equivalent to the given controller.

2. *Choice of homomorphic cryptosystem and application to dynamic operation.* The employed cryptosystem should be appropriate to the dynamic operation of the controller. If the implemented controller is to perform recursive multiplication by an encrypted parameter for each time iteration (just as in (4)), the cryptosystem should be homomorphic with respect to such recursive operation. To that end, it can be expected that the encryption should be homomorphic with respect to both addition and multiplication, and it should allow recursive multiplication as encrypted, for an unlimited number of times.

Let us begin with the second issue in the next subsection. We introduce a Learning With Errors (LWE) based encryption called "GSW-LWE" proposed in Chillotti et al. (2016), which can be understood as a combination of "Gentry–Sahai–Waters (GSW)" encryption scheme (Gentry, Sahai, & Waters, 2013) and a basic LWE-based encryption (Lindner & Peikert, 2011). Detailed explanations on the encryption scheme and algorithms will be followed by a discussion on its benefits to dynamic operation.

3.1. An Learning With Errors-based cryptosystem for encrypted recursive multiplication

We first introduce a basic version of LWE-based cryptosystem.

3.1.1. Encryption method, security, and additively homomorphic property (*Lindner & Peikert, 2011; Regev, 2009*)

Let the set $\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$, $q \in \mathbb{N}$, be the space of messages for encryption. We have seen in Section 2.3 and Fig. 5 that "adding a random number $r \in \mathbb{Z}_q$ and taking modulo operation by q" is an easy way to conceal a message $m \in \mathbb{Z}_q$. Though it is quite simple, it may be impractical in the sense that the information of the number $r \in \mathbb{Z}_q$ is required for restoring the message, which is newly sampled whenever a message is encrypted.

To be a cryptosystem using the same "key" for each encryption, LWE based schemes generate "random-like" numbers from a fixed portion of data, called an encryption key; let a row vector $\mathsf{sk} = [\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_n] \in \mathbb{Z}^n$, $n \in \mathbb{N}$, be chosen, which we will call the encryption key or the secret key. Then, the idea of LWE-based schemes is that the number $c \in \mathbb{Z}_q$, generated as

$$c = \begin{bmatrix} \mathsf{sk}_1 & \mathsf{sk}_2 & \cdots & \mathsf{sk}_n \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} + e \mod q \tag{5}$$

$$= \mathsf{sk} \cdot a + e \mod q,$$

looks "almost uniformly random" in the space $\mathbb{Z}_q$, where the components of $a = [a_1, a_2, \ldots, a_n]^\top$ are random numbers in $\mathbb{Z}_q$ (i.e., sampled from the uniform distribution over $\mathbb{Z}_q$), and $e \in \mathbb{Z}$ is a "small error" sampled from $N(0, \sigma)$, which denotes the zero-mean discrete Gaussian distribution with standard deviation $\sigma > 0$.

The claim that the method (5) generates "almost random" numbers is in fact exactly what the Learning With Errors (LWE) problem considers. Let a "sufficient" amount of samples $\{c_i\}_{i=1}^N$ in $\mathbb{Z}_q$ be generated using the key $\mathsf{sk} \in \mathbb{Z}^n$, as

$$\begin{aligned} c_1 &= \mathsf{sk} \cdot \vec{a}_1 + e_1 \mod q \\ c_2 &= \mathsf{sk} \cdot \vec{a}_2 + e_2 \mod q \\ &\vdots \\ c_N &= \mathsf{sk} \cdot \vec{a}_N + e_N \mod q, \end{aligned} \tag{6}$$

where the components of $a_i \in \mathbb{Z}_q^n$ are uniformly sampled from $\mathbb{Z}_q$ and $e_i \in \mathbb{Z}$ is sampled from the distribution $N(0, \sigma)$, for each $i = 1, \ldots, N$. Then, given the information of $\{c_i\}_{i=1}^N$ and $\{\vec{a}_i\}_{i=1}^N$, the LWE problem is to find out if the data $\{c_i\}_{i=1}^N$ are generated through the process (6), or are just uniformly generated random numbers. If this problem is hard to solve, it implies that the information of the secret key sk is "secure" from unauthorized parties receiving or eavesdropping $\{\vec{a}_i\}_{i=1}^N$ and $\{c_i\}_{i=1}^N$ only. And, another version of LWE problem is to solve Eq. (6) and find the value of the key sk, from given data of $\{c_i\}_{i=1}^N$ and $\{\vec{a}_i\}_{i=1}^N$.

Note that Eq. (6) is defined over the space $\mathbb{Z}_q$ with modulo operation, so that the problem is different from the simple least square problem over real numbers. It has been known that the LWE-problems are hard; for example, it has been proven in Regev (2009) that the latter version of problem is hard as "worst-case lattice problems", and it has also been known as "post-quantum cryptography" (Chen et al., 2016).

The level of "hardness" is determined by the parameters; the dimension n for the key $\mathsf{sk} \in \mathbb{Z}^n$ and the vector $a \in \mathbb{Z}_q^n$, the modulus q, and the standard deviation σ for the error distribution $N(0, \sigma)$. For example, in Lindner and Peikert (2011), it has been suggested to choose the parameters to satisfy

$$n \log q \geq \frac{\lambda + 110}{7.2} \cdot \left(\log \left(\frac{\sqrt{2\pi}\sigma}{q} \right) \right)^2, \tag{7}$$

in order for "λ-bit security", which means more than 2^λ-times of iterations of a certain computation are required to solve the LWE problem. Typically, to ensure a higher level of security, a larger dimension n is chosen for the key $\mathsf{sk} \in \mathbb{Z}^n$.

Then, thanks to the security guaranteed from the LWE problems, we may define an encrypted algorithm with the key[3] $\mathsf{sk} \in \mathbb{Z}^n$, as

$$\text{for } m \in \mathbb{Z}_q, \quad \mathsf{Enc}(m) := \begin{bmatrix} m + \mathsf{sk} \cdot a + e \\ a \end{bmatrix} \mod q \quad \in \mathbb{Z}_q^{\mathfrak{n}}, \qquad \mathfrak{n} := n + 1, \tag{8}$$

where the components of the column vector $a \in \mathbb{Z}_q^n$ are uniformly sampled from $\mathbb{Z}_q$ and the error $e \in \mathbb{Z}$ is sampled from $N(0, \sigma)$ analogous to (5). We omit the argument of the key sk from all the cryptosystem algorithms, for simplicity. Note that, the encryption outcome becomes an $(\mathfrak{n} = n + 1)$-dimensional vector consisting of elements in $\mathbb{Z}_q$; as LWE problems suppose that the vector a be public, so that it can be treated as a part of encrypted message (it will be used for decryption, to restore the message). It can be understood that every encrypted message consists of "message part", "random-like part", and "error part", as

$$\mathsf{Enc}(m) = \begin{bmatrix} m \\ 0_{n \times 1} \end{bmatrix} + \begin{bmatrix} \mathsf{sk} \cdot a \\ a \end{bmatrix} + \begin{bmatrix} e \\ 0_{n \times 1} \end{bmatrix} \mod q, \tag{9}$$

where the information of the message is contained in the first component.

As the encryption is to add a random-like number $\mathsf{sk} \cdot a$ to the message together with an error, it can be decrypted by simply canceling

---

[3] Usually, the components of sk are generated by sampling from the distribution $N(0, \sigma)$, the same distribution for the errors.

out the same number $\text{sk} \cdot a$; let the decryption for a ciphertext $\mathbf{c} \in \mathbb{Z}_q^n$ be defined as

$$\text{Dec}(\mathbf{c}) := \begin{bmatrix} 1 & -\text{sk} \end{bmatrix} \cdot \mathbf{c} \mod q \quad \in \mathbb{Z}_q, \tag{10}$$

considering that the "random-like term" $[\text{sk} \cdot a, a^\top]^\top$ in (9) as a column vector will be canceled out whenever the secret key $[1, -\text{sk}]$ is multiplied from the left, regardless of the random part a. Indeed, it is obvious that

$$\text{Dec}(\text{Enc}(m)) = m + e \mod q \tag{11}$$

holds, when the ciphertext c is simply a newly encrypted message. The cryptosystem being described is "symmetric" as a basic case, where the same key sk is used for both the encryption and decryption. For a public-key LWE-based cryptosystem in which encryption is possible without knowledge of the key sk, interested readers may refer to Lindner and Peikert (2011).

It can be seen in (11) that the error e, "injected" during encryption because of security, may perturb the decryption outcome not equal to the original message. To negate the effect of errors, a scale factor can be used for the messages; let us neglect the probability that a sample from the distribution $N(0, \sigma)$ is larger than a particular multiple $n_0\sigma$ of σ, $n_0 \in \mathbb{N}$, and assume that every error e sampled from $N(0, \sigma)$ is such that $|e| \le n_0\sigma$. Then, by modifying the encryption as

$$\text{Enc}_\text{L}(m) := \text{Enc}(\text{L}m \mod q), \tag{12}$$

with some scale factor $\text{L} \in \mathbb{N}$ such that $\text{L}/2 > n_0\sigma$, the message can be restored without error as

$$\text{Dec}(\text{Enc}_\text{L}(m)) = \text{L}m + e \mod q \implies \left\lceil \frac{\text{L}m + e \mod q}{\text{L}} \right\rfloor = m, \tag{13}$$

as long as $0 < m < (q/\text{L}) - (1/2)$ so that $0 \le \text{L}m + e < q$.

As one of the simplest versions of LWE based schemes, it can be easily seen that the encryption Enc is homomorphic with respect to addition. Let the addition of two encrypted messages be defined simply by the component-wise modular addition as

$$\mathbf{c}_1 \in \mathbb{Z}_q^n, \quad \mathbf{c}_2 \in \mathbb{Z}_q^n \quad \mapsto \quad \mathbf{c}_1 + \mathbf{c}_2 \mod q \quad \in \mathbb{Z}_q^n.$$

Then, since the decryption Dec is nothing but multiplying a row vector, by the distributive law, it is obvious that

$$\begin{aligned} \text{Dec}(\mathbf{c}_1 + \mathbf{c}_2 \mod q) &= \begin{bmatrix} 1 & -\text{sk} \end{bmatrix} \cdot (\mathbf{c}_1 + \mathbf{c}_2) \mod q \\ &= \text{Dec}(\mathbf{c}_1) + \text{Dec}(\mathbf{c}_2) \mod q \end{aligned} \tag{14}$$

holds for every ciphertexts $\mathbf{c}_1 \in \mathbb{Z}_q^n$ and $\mathbf{c}_2 \in \mathbb{Z}_q^n$. It means that the modular addition for the messages in $\mathbb{Z}_q$ can be performed as encrypted, by the same (component-wise) modular addition over the ciphertext space $\mathbb{Z}_q^n$, because the decryption of the computation outcome matches to the addition of the plaintext messages $\text{Dec}(\mathbf{c}_1)$ and $\text{Dec}(\mathbf{c}_2)$.

Multiplication of a ciphertext $\mathbf{c}_1 \in \mathbb{Z}_q^n$ by an un-encrypted integer $k \in \mathbb{Z}$ can be done analogously, as

$$k \in \mathbb{Z}, \quad \mathbf{c}_1 \in \mathbb{Z}_q^n \quad \mapsto \quad k \cdot \mathbf{c}_1 \mod q \quad \in \mathbb{Z}_q^n, \tag{15}$$

which is simply the component-wise modular multiplication by the constant k. It is also clear that

$$\begin{aligned} \text{Dec}(k \cdot \mathbf{c}_1 \mod q) &= \begin{bmatrix} 1 & -\text{sk} \end{bmatrix} \cdot (k \cdot \mathbf{c}_1) \mod q \\ &= k \cdot \text{Dec}(\mathbf{c}_1) \mod q, \end{aligned} \tag{16}$$

so the decryption outcome of integer multiplication for encrypted messages matches to that over plaintexts.

In general, LWE-based cryptosystems are also multiplicatively homomorphic, which means they allow multiplication by an encrypted number to ciphertexts as well as by the un-encrypted integer multiplication as in (15). Nonetheless, in fact, there is no "trivial way" of performing multiplication of a ciphertext $\mathbf{c}_1 \in \mathbb{Z}_q^n$ to another ciphertext $\mathbf{c}_2 \in \mathbb{Z}_q^n$, since the component-wise multiplication over $\mathbb{Z}_q^n$ would not work (i.e., decryption outcome of component-wise multiplication would not match with multiplication between the messages), and it

seems there is no other "intuitive way" of multiplying two vectors in $\mathbb{Z}_q^n$ yielding the same form in $\mathbb{Z}_q^n$. In this regard, a separate encryption method as well as an encrypted multiplication method proposed in Chillotti et al. (2016) and Gentry et al. (2013) is introduced in the next subsection.

3.1.2. Gentry-Sahai-Waters (GSW) scheme and encrypted recursive multiplication (Chillotti et al., 2016; Gentry et al., 2013)

An idea for multiplication by an encrypted message is to consider a separate encryption method whose outcome is a matrix in $\mathbb{Z}_q^{n \times n}$, so that it can be multiplied to the ciphertexts of column vectors in $\mathbb{Z}_q^n$ from the left naturally; consider a "multiplier" $k \in \mathbb{Z}_q$ encrypted as

$$\begin{aligned} k \in \mathbb{Z}_q \quad \mapsto \quad \mathbf{C} = k \cdot I_n &+ \begin{bmatrix} \text{sk} \cdot a_1 & \text{sk} \cdot a_2 & \cdots & \text{sk} \cdot a_n \\ a_1 & a_2 & \cdots & a_n \end{bmatrix} \\ &+ \begin{bmatrix} e_1 & e_2 & \cdots & e_n \\ 0_{n \times 1} & 0_{n \times 1} & \cdots & 0_{n \times 1} \end{bmatrix} \mod q \quad \in \mathbb{Z}_q^{n \times n} \end{aligned} \tag{17}$$

where $a_i \in \mathbb{Z}_q^n$ is a uniformly random vector and e_i is an error sampled from $N(0, \sigma)$ for each $i = 1, \dots n$, respectively. The message part $k \cdot I_n$ in (17) is added by a matrix consisting of "random-like" vectors plus errors analogous to (9), so obviously it is also an LWE-based encryption.

Then, it can be checked if the multiplication outcome of

$$\mathbf{C} \in \mathbb{Z}_q^{n \times n}, \quad \mathbf{c} \in \mathbb{Z}_q^n \quad \mapsto \quad \mathbf{C} \cdot \mathbf{c} \mod q \quad \in \mathbb{Z}_q^n \tag{18}$$

(multiplication of the matrix $\mathbf{C} \in \mathbb{Z}_q^{n \times n}$ from (17) to a ciphertext vector $\mathbf{c} \in \mathbb{Z}_q^n$) has the message parts multiplied with each other; let us decrypt the outcome (18) by multiplying $[1, -\text{sk}]$ from the left (just as in (10)), which yields

$$\begin{aligned} &\text{Dec}(\mathbf{C} \cdot \mathbf{c} \mod q) \\ &= \begin{bmatrix} 1 & -\text{sk} \end{bmatrix} \cdot \left(k \cdot I_n + \begin{bmatrix} \text{sk} \cdot a_1 & \cdots & \text{sk} \cdot a_n \\ a_1 & \cdots & a_n \end{bmatrix} \right. \\ &\quad \left. + \begin{bmatrix} e_1 & \cdots & e_n \\ 0_{n \times 1} & \cdots & 0_{n \times 1} \end{bmatrix} \right) \cdot \mathbf{c} \mod q \\ &= \begin{bmatrix} 1 & -\text{sk} \end{bmatrix} \cdot \left(k \cdot I_n + \begin{bmatrix} e_1 & \cdots & e_n \\ 0_{n \times 1} & \cdots & 0_{n \times 1} \end{bmatrix} \right) \cdot \mathbf{c} \mod q \\ &= k \cdot \text{Dec}(\mathbf{c}) + \begin{bmatrix} e_1 & \cdots & e_n \end{bmatrix} \cdot \mathbf{c} \mod q. \end{aligned}$$

It can be seen that the encryption is homomorphic with respect to the multiplication, in the sense that the decryption outcome has the message $\text{Dec}(\mathbf{c})$ multiplied by k. However, it can be found that the injected errors $[e_1, \dots, e_n]$, whose components were smaller than $n_0\sigma$, have been multiplied by the ciphertext $\mathbf{c} \in \mathbb{Z}_q^n$ consisting of large numbers. Then, the size of the error "grows" too large and it will even dominate the message part, so that the computation result will not be correct at all. Thus, the described method cannot be applied as it is, so modified as follows.

Considering that the problem of the method (18) was due to the large size of the "multiplicand" $\mathbf{c} \in \mathbb{Z}_q^n$ amplifying the errors, the method in Gentry et al. (2013) called "Gentry–Sahai–Waters (GSW)" scheme suggests that the multiplication be done in a different manner, to reduce the size of the multiplicand; consider multiplication of a vector $\mathbf{x} \in \mathbb{Z}_q^n$ by $k \in \mathbb{Z}_q$, represented as

$$k \cdot \mathbf{x} = k \cdot \left(\sum_{i=0}^{d-1} v^i \cdot \mathbf{x}_i \right) = k \cdot \begin{bmatrix} I_n & v \cdot I_n & \cdots & v^{d-1} \cdot I_n \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{d-1} \end{bmatrix}, \tag{19}$$

where $v \in \mathbb{N}$ is a base chosen such that $v^{d-1} < q \le v^d$ with some $d \in \mathbb{N}$, so that the vector $\mathbf{x} \in \mathbb{Z}_q^n$ is represented by "v-ary numeral system" as $\mathbf{x} = \sum_{i=0}^{d-1} v^i \cdot \mathbf{x}_i$ with some vectors $\{\mathbf{x}_i\}_{i=0}^{d-1}$ of non-negative integers such

that $\|x_i\| < \nu$, $\forall i$. Then, by defining

$$G := \begin{bmatrix} I_{\mathfrak{n}} & \nu \cdot I_{\mathfrak{n}} & \cdots & \nu^{d-1} \cdot I_{\mathfrak{n}} \end{bmatrix} \quad \text{and} \quad D(\mathsf{x}) := \begin{bmatrix} \mathsf{x}_0 \\ \mathsf{x}_1 \\ \vdots \\ \mathsf{x}_{d-1} \end{bmatrix},$$

and rewriting (19) as

$$k \cdot \mathsf{x} = (k \cdot G) \cdot D(\mathsf{x}),$$

one can have the size of the multiplicand reduced as $\|D(\mathsf{x})\| < \nu$, thanks to the "decomposition" function D. For example, consider the equation $77 = 7 \times 11$ re-written as

$$77 = \left(7 \times \begin{bmatrix} 2^0 & 2^1 & 2^2 & 2^3 \end{bmatrix}\right) \times \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

with $d = 4$, $\nu = 2$. It can be seen that the norm of the multiplicand (which was 11) is reduced.

From this observation, the GSW scheme slightly modifies the encryption algorithm (17), as

$$\mathrm{Enc}'(k) := k \cdot G + \begin{bmatrix} \mathsf{sk} \cdot a_1 & \mathsf{sk} \cdot a_2 & \cdots & \mathsf{sk} \cdot a_{d\mathfrak{n}} \\ a_1 & a_2 & \cdots & a_{d\mathfrak{n}} \end{bmatrix}$$
$$+ \begin{bmatrix} e_1 & e_2 & \cdots & e_{d\mathfrak{n}} \\ 0_{n\times 1} & 0_{n\times 1} & \cdots & 0_{n\times 1} \end{bmatrix} \mod q \quad \in \mathbb{Z}_q^{\mathfrak{n}\times d\mathfrak{n}},$$

by substituting $I_{\mathfrak{n}}$ in (17) by the matrix G and increasing the column dimension d-times. Now, let the multiplication of a ciphertext $\mathbf{c} \in \mathbb{Z}_q^{\mathfrak{n}}$ by an encrypted message $\mathrm{Enc}'(k)$, be performed as

$$\mathrm{Enc}'(k) \in \mathbb{Z}_q^{\mathfrak{n}\times d\mathfrak{n}}, \quad \mathbf{c} \in \mathbb{Z}_q^{\mathfrak{n}} \quad \mapsto \quad \mathrm{Enc}'(k) \cdot D(\mathbf{c}) \mod q \in \mathbb{Z}_q^{\mathfrak{n}}, \quad (20)$$

which enlarges the dimension of $\mathbf{c}$ by the decomposition first and then multiply to the encrypted multiplier. Then, let us decrypt the outcome and check the multiplicatively homomorphic property and the growth of the error again, as

$$\mathrm{Dec}(\mathrm{Enc}'(k) \cdot D(\mathbf{c}) \mod q)$$
$$= \begin{bmatrix} 1 & -\mathsf{sk} \end{bmatrix} \cdot \left(k \cdot G + \begin{bmatrix} \mathsf{sk} \cdot a_1 & \cdots & \mathsf{sk} \cdot a_{d\mathfrak{n}} \\ a_1 & \cdots & a_{d\mathfrak{n}} \end{bmatrix} \right.$$
$$+ \left. \begin{bmatrix} e_1 & \cdots & e_{d\mathfrak{n}} \\ 0_{n\times 1} & \cdots & 0_{n\times 1} \end{bmatrix} \right) \cdot D(\mathbf{c}) \mod q$$
$$= \begin{bmatrix} 1 & -\mathsf{sk} \end{bmatrix} \cdot \left(k \cdot G + \begin{bmatrix} e_1 & \cdots & e_{d\mathfrak{n}} \\ 0_{n\times 1} & \cdots & 0_{n\times 1} \end{bmatrix} \right) \cdot D(\mathbf{c}) \mod q$$
$$= \begin{bmatrix} 1 & -\mathsf{sk} \end{bmatrix} \cdot k \cdot \mathbf{c} + \begin{bmatrix} e_1 & \cdots & e_{d\mathfrak{n}} \end{bmatrix} \cdot D(\mathbf{c}) \mod q$$
$$= k \cdot \mathrm{Dec}(\mathbf{c}) + \begin{bmatrix} e_1 & \cdots & e_{d\mathfrak{n}} \end{bmatrix} \cdot D(\mathbf{c}) \mod q.$$

It turns out that, thanks to the size of the multiplicand $\mathbf{c} \in \mathbb{Z}_q^{\mathfrak{n}}$ reduced by the decomposition D as $\|D(\mathbf{c})\| < \nu$, the effect of the error becomes bounded by a constant proportional to the parameter ν (recall that the base ν can be chosen such that $\nu \ll q$). Putting all together, the homomorphic properties of the described cryptosystem is listed as the following proposition.

Proposition 1. *The following holds.*

1. *For every $m \in \mathbb{Z}_q$, it satisfies $\mathrm{Dec}(\mathrm{Enc}(m)) = m + e \mod q$, with some $e \in \mathbb{Z}$ such that $|e| \le n_0 \sigma$.*
2. *For any ciphertexts $\mathbf{c}_1 \in \mathbb{Z}_q^{\mathfrak{n}}$ and $\mathbf{c}_2 \in \mathbb{Z}_q^{\mathfrak{n}}$, they satisfy $\mathrm{Dec}(\mathbf{c}_1 + \mathbf{c}_2 \mod q) = \mathrm{Dec}(\mathbf{c}_1) + \mathrm{Dec}(\mathbf{c}_2) \mod q$.*
3. *For any $k \in \mathbb{Z}$ and $\mathbf{c} \in \mathbb{Z}_q^{\mathfrak{n}}$, they satisfy $\mathrm{Dec}(k \cdot \mathbf{c} \mod q) = k \cdot \mathrm{Dec}(\mathbf{c}) \mod q$.*
4. *For any $k \in \mathbb{Z}_q$ and $\mathbf{c} \in \mathbb{Z}_q^{\mathfrak{n}}$, they satisfy $\mathrm{Dec}(\mathrm{Enc}'(k) \cdot D(\mathbf{c}) \mod q) = k \cdot \mathrm{Dec}(\mathbf{c}) + e \mod q$, with some $e \in \mathbb{Z}$ such that $|e| \le \Delta_{\mathrm{Mult}} := d\mathfrak{n} \cdot n_0 \sigma \cdot \nu$.* □

The homomorphic properties directly extended to matrix–vector multiplication over encrypted data, by considering component-wise encryption and operations; let us abuse the notation and the algorithms Enc and Dec be also applied to vector of messages component-wisely, as

$$\mathrm{Enc}(\vec{m}) := \begin{bmatrix} \mathrm{Enc}(m_1) \\ \mathrm{Enc}(m_2) \\ \vdots \\ \mathrm{Enc}(m_l) \end{bmatrix} \in \mathbb{Z}_q^{l\mathfrak{n}}, \quad (21a)$$

for $\vec{m} = \mathrm{col}\{m_i\}_{i=1}^l \in \mathbb{Z}_q^l$, and

$$\mathrm{Dec}(\vec{\mathbf{c}}) := \begin{bmatrix} \mathrm{Dec}(\mathbf{c}_1) \\ \mathrm{Dec}(\mathbf{c}_2) \\ \vdots \\ \mathrm{Dec}(\mathbf{c}_l) \end{bmatrix} \in \mathbb{Z}_q^l, \quad (21b)$$

for $\vec{\mathbf{c}} = \mathrm{col}\{\mathbf{c}_i\}_{i=1}^l \in \mathbb{Z}_q^{l\mathfrak{n}}$, $\mathbf{c}_i \in \mathbb{Z}_q^{\mathfrak{n}}$, $i = 1, \ldots, l$. And, let matrices consisting of elements in $\mathbb{Z}_q$ be encrypted using the encryption Enc' component-wisely, as

$$\mathrm{Enc}'(F) = \begin{bmatrix} \mathrm{Enc}'(F_{11}) & \mathrm{Enc}'(F_{12}) & \cdots & \mathrm{Enc}'(F_{1l_2}) \\ \mathrm{Enc}'(F_{21}) & \mathrm{Enc}'(F_{22}) & \cdots & \mathrm{Enc}'(F_{2l_2}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Enc}'(F_{l_1 1}) & \mathrm{Enc}'(F_{l_1 2}) & \cdots & \mathrm{Enc}'(F_{l_1 l_2}) \end{bmatrix} \in \mathbb{Z}_q^{(l_1\mathfrak{n})\times(dl_2\mathfrak{n})} \quad (21c)$$

for $F = [F_{ij}] \in \mathbb{Z}_q^{l_1 \times l_2}$, and let the multiplication of a vector of encrypted messages by an encrypted matrix be considered as

$$\mathrm{Enc}'(F) \in \mathbb{Z}_q^{(l_1\mathfrak{n})\times(dl_2\mathfrak{n})}, \quad \vec{\mathbf{c}} \in \mathbb{Z}_q^{l_2\mathfrak{n}} \quad \mapsto \quad \mathrm{Enc}'(F) \cdot D(\vec{\mathbf{c}}) \mod q, \quad (21d)$$

where we also let the decomposition D be applied to a vector of encrypted messages component-wisely as $D(\vec{\mathbf{c}}) := \mathrm{col}\{D(\mathbf{c}_i)\}_{i=1}^{l_2} \in \mathbb{Z}_q^{l_2 d\mathfrak{n}}$. Then, it is easy to verify that the following proposition holds.

Proposition 2. *For any matrix $F \in \mathbb{Z}_q^{l_1 \times l_2}$ and a ciphertext of encrypted messages $\vec{\mathbf{c}} \in \mathbb{Z}_q^{l_2\mathfrak{n}}$, they satisfy[4]*

$$\mathrm{Dec}(\mathrm{Enc}'(F) \cdot D(\vec{\mathbf{c}}) \mod q) = F \cdot \mathrm{Dec}(\vec{\mathbf{c}}) + \vec{e} \mod q, \quad (22)$$

with some $\vec{e} \in \mathbb{Z}^{l_1}$ such that $\|\vec{e}\| \le l_2 \Delta_{\mathrm{Mult}}$. □

Note that the multiplication of an "LWE type" ciphertext by a "GSW type" encrypted matrix yields an "LWE type" ciphertext. The encryption method of Enc' has been introduced in Gentry et al. (2013) where multiplication between GSW type ciphertexts has been presented, and the multiplication of an LWE type ciphertext by a GSW type ciphertext has been considered in Chillotti et al. (2016).

Now, we further discuss homomorphic property with respect to "recursive" operation. Let us compare the following two types of homomorphic properties:

Property 1 : $\mathrm{Dec}(\mathrm{Enc}(x_1) * \mathrm{Enc}(x_2)) = x_1 \cdot x_2 \mod q$,

              $\forall x_1 \in \mathbb{Z}_q$, $x_2 \in \mathbb{Z}_q$ (plaintexts)

Property 2 : $\mathrm{Dec}(\mathbf{c}_1 * \mathbf{c}_2) = \mathrm{Dec}(\mathbf{c}_1) \cdot \mathrm{Dec}(\mathbf{c}_2) \mod q$,

              $\forall \mathbf{c}_1 \in \mathbb{Z}_q^{\mathfrak{n}}$, $\mathbf{c}_2 \in \mathbb{Z}_q^{\mathfrak{n}}$ (ciphertexts)

where $*$ denotes a certain operation over encrypted data that matches to the multiplication over plaintexts. Each of the properties implies an ability of multiplication over encrypted data, but it can be observed that Property 2 implies Property 1, but Property 1 does not imply Property 2 conversely, in general; this is because Property 1 can be applied to "newly" encrypted messages only, whereas Property 2 can be applied to *any* ciphertexts. For example, for a ciphertext $\mathbf{c} \in \mathbb{Z}_q^{\mathfrak{n}}$ which is a computation outcome as $\mathbf{c} = \mathrm{Enc}(x_1) * \mathrm{Enc}(x_2) \mod q$ with some $x_1 \in \mathbb{Z}_q$ and $x_2 \in \mathbb{Z}_q$, it is clear that the property of multiplication is applicable

---

[4]   Note that Δ_{Mult} is defined from Proposition 1. 4.

to **c** if the cryptosystem satisfies Property 2, whereas the multiplication may not be applicable to **c** even if the cryptosystem satisfies Property 1, because there is no guarantee that there exists a message $x' \in \mathbb{Z}_q$ such that $\mathbf{c} = \mathsf{Enc}(x')$.

Keeping this observation in mind, let us revisit the property (22) that we obtained. In the left hand side where the encrypted multiplication is performed, the argument of left multiplier considers GSW type ciphertext $\mathsf{Enc}'(F)$ as newly encrypted message only, but the argument of the right multiplicand considers any LWE type ciphertext, regardless if is a newly encrypted message or it was used for certain encrypted operations. Since the outcome of the multiplication is also an LWE type ciphertext, the property (22) implies that the encrypted multiplication is applicable to LWE type ciphertexts *recursively, unlimited number of times*. Meanwhile, since an error $\vec{e}$ is added to the encrypted message whenever the multiplication is performed, it is notable that the "growth of error" can be accumulated under recursive operation. In the next subsection, when the LWE based cryptosystem is applied to dynamic control operations, it will be seen that the growth of error accumulated in LWE type ciphertexts under recursive operation can be "controlled" under stability.

The effect of error can be reduced by using a scale factor for the LWE type encryption, as seen in (13); for instance, let the property (22) be rewritten with respect to the encryption (12), as

$$\mathsf{Dec}(\mathsf{Enc}'(F) \cdot D(\mathsf{Enc}_\mathsf{L}(x))) \mod q) = F(L \cdot x + \vec{e}_1) + \vec{e}_2 \mod q,$$

where $x \in \mathbb{Z}_q^{l_2}$, the error $\vec{e}_1 \in \mathbb{Z}_q^{l_2}$ from the encryption Enc_L is such that $\|\vec{e}_1\| \leq n_0\sigma$, and the error $\vec{e}_2 \in \mathbb{Z}_q^{l_1}$ from the multiplication is such that $\|\vec{e}_2\| \leq l_2\Delta_\mathsf{Mult}$. Then, by choosing the parameter L considering the size of grown error such that $L/2 > \|F\| \cdot n_0\sigma + l_2\Delta_\mathsf{Mult}$, the expected outcome $F \cdot x$ can be obtained without the error, as

$$\left\lceil \frac{F \cdot (Lx + \vec{e}_1) + \vec{e}_2 \mod q}{L} \right\rfloor = F \cdot x, \tag{23}$$

as long as all the components of $F \cdot x$ are less than $(q/L) - (1/2)$ and greater than 0 so that the modulo operation in (23) does nothing about the argument value. Note that the scale factor L is used for the LWE type encryption only and not used for GSW type encryption, so that the scaled LWE type messages are not multiplied with another scale factor through the multiplication by GSW type encrypted messages.

As a result of the enabled recursive multiplication, a class of dynamic systems, which can be implemented to run over encrypted data exploiting the homomorphic properties, is specified. Consider a system defined over the space $\mathbb{Z}_q$ written as

$$z(t + 1) = F \cdot z(t) + G \cdot y(t) \mod q$$
$$u(t) = H \cdot z(t) + J \cdot y(t) \mod q \tag{24}$$
$$z(0) = z_0 \in \mathbb{Z}_q^\ell,$$

where $z(t) \in \mathbb{Z}_q^\ell$ is the state with the initial value z_0, $y(t) \in \mathbb{Z}_q^p$ is the input, $u(t) \in \mathbb{Z}_q^m$ is the output, and $\{F, G, H, J\}$ are matrices consisting of elements in $\mathbb{Z}_q$. Then, an implication of Proposition 2 is that the dynamic operation of (24) can be performed over encrypted data, as

$$\mathbf{z}(t + 1) = \mathbf{F} \cdot D(\mathbf{z}(t)) + \mathbf{G} \cdot D(\mathsf{Enc}(y(t))) \mod q$$
$$\mathbf{u}(t) = \mathbf{H} \cdot D(\mathbf{z}(t)) + \mathbf{J} \cdot D(\mathsf{Enc}(y(t))) \mod q \tag{25}$$
$$\mathbf{z}(0) = \mathsf{Enc}(z_0),$$

where $\mathbf{z}(t) \in \mathbb{Z}_q^{\ell n}$ is the state with the initial value $\mathsf{Enc}(z_0)$, $\mathsf{Enc}(y(t)) \in \mathbb{Z}_q^{pn}$ is the input, $\mathbf{u}(t) \in \mathbb{Z}_q^{mn}$ is the output, and $\{\mathbf{F}, \mathbf{G}, \mathbf{H}, \mathbf{J}\}$ are the encryptions of the matrices from (24), as

$$\mathbf{F} = \mathsf{Enc}'(F), \quad \mathbf{G} = \mathsf{Enc}'(G), \quad \mathbf{H} = \mathsf{Enc}'(H), \quad \mathbf{J} = \mathsf{Enc}'(J).$$

As the configuration of an encrypted system described in Fig. 2(b), it stores the encrypted parameters $\{\mathbf{F}, \mathbf{G}, \mathbf{H}, \mathbf{J}, \mathbf{z}_0\}$, receives the input $y(t)$ as an LWE type newly encrypted signal, and computes the next state $\mathbf{z}(t + 1)$ and the output $\mathbf{u}(t)$, respectively, using the GSW-LWE matrix

multiplication and the addition between LWE type ciphertexts. The performance of the system (25) can be analyzed, by decrypting the signals $\mathbf{z}(t)$ and $\mathbf{u}(t)$ and comparing to that of the un-encrypted model (24), as the following proposition.

Proposition 3. *Consider the messages $\bar{z}(t) := \mathsf{Dec}(\mathbf{z}(t)) \in \mathbb{Z}_q^\ell$ and $\bar{u}(t) := \mathsf{Dec}(\mathbf{u}(t)) \in \mathbb{Z}_q^m$ of the encrypted trajectories of (25). They obey*

$$\bar{z}(t + 1) = F \cdot \bar{z}(t) + G \cdot (y(t) + \Delta_y(t)) + \Delta_z(t) \mod q$$
$$\bar{u}(t) = H \cdot \bar{z}(t) + J \cdot (y(t) + \Delta_y(t)) + \Delta_u(t) \mod q \tag{26}$$
$$\bar{z}(0) = z_0 + \Delta_0 \mod q,$$

with some $\Delta_y(t) \in \mathbb{Z}^p$, $\Delta_z(t) \in \mathbb{Z}^\ell$, $\Delta_u(t) \in \mathbb{Z}^m$, and $\Delta_0 \in \mathbb{Z}^\ell$ such that

$$\|\Delta_y(t)\| \leq n_0\sigma, \qquad \|\Delta_z(t)\| \leq (\ell + p)\Delta_\mathsf{Mult},$$
$$\|\Delta_u(t)\| \leq (\ell + p)\Delta_\mathsf{Mult}, \qquad \|\Delta_0\| \leq n_0\sigma, \tag{27}$$

respectively. □

Proof. By Propositions 2 and 1. 1, it is obvious that

$$\begin{bmatrix} \bar{z}(t+1) \\ \bar{u}(t) \end{bmatrix} = \mathsf{Dec}\left(\begin{bmatrix} \mathbf{z}(t+1) \\ \mathbf{u}(t) \end{bmatrix}\right) = \mathsf{Dec}\left(\begin{bmatrix} \mathbf{F} & \mathbf{G} \\ \mathbf{H} & \mathbf{J} \end{bmatrix} \cdot D\left(\begin{bmatrix} \mathbf{z}(t) \\ \mathsf{Enc}(y(t)) \end{bmatrix}\right)\right)$$
$$= \begin{bmatrix} F & G \\ H & J \end{bmatrix} \cdot \begin{bmatrix} \bar{z}(t) \\ y(t) + \Delta_y(t) \end{bmatrix} + \begin{bmatrix} \Delta_z(t) \\ \Delta_u(t) \end{bmatrix} \mod q$$

and $\mathsf{Dec}(\mathbf{z}(0)) = \mathsf{Dec}(\mathsf{Enc}(z_0)) = z_0 + \Delta_0$ hold, with some $\{\Delta_y(t), \Delta_z(t), \Delta_u(t), \Delta_0\}$ satisfying (27). ∎

Proposition 3 shows that the performance of the encrypted system (25) is equivalent to the system (26), which has the same parameters $\{F, G, H, J, z_0\}$ with that of (24). It is found that the effect of the errors is denoted by $\{\Delta_z(t), \Delta_u(t), \Delta_0\}$, which can be seen as disturbances or perturbations bounded as (27).

In order to have the error effect relatively small, the encryption Enc_L with the scale factor L can be used for the input $y(t)$ and the state $z(t)$, as

$$\mathbf{z}(t + 1) = \mathbf{F} \cdot D(\mathbf{z}(t)) + \mathbf{G} \cdot D(\mathsf{Enc}_\mathsf{L}(y(t))) \mod q$$
$$\mathbf{u}(t) = \mathbf{H} \cdot D(\mathbf{z}(t)) + \mathbf{J} \cdot D(\mathsf{Enc}_\mathsf{L}(y(t))) \mod q \tag{28}$$
$$\mathbf{z}(0) = \mathsf{Enc}_\mathsf{L}(z_0).$$

In the following corollary, it can be observed that only the messages in the dynamics are scaled by the factor L, while the bound for the errors remains the same.

Corollary 1. *The messages $\bar{z}(t) = \mathsf{Dec}(\mathbf{z}(t))$ and $\bar{u}(t) = \mathsf{Dec}(\mathbf{u}(t))$ of the system (28) obey*

$$\bar{z}(t + 1) = F \cdot \bar{z}(t) + G \cdot (L \cdot y(t) + \Delta_y(t)) + \Delta_z(t) \mod q$$
$$\bar{u}(t) = H \cdot \bar{z}(t) + J \cdot (L \cdot y(t) + \Delta_y(t)) + \Delta_u(t) \mod q \tag{29}$$
$$\bar{z}(0) = L \cdot z_0 + \Delta_0 \mod q,$$

where the errors $\Delta_y(t) \in \mathbb{Z}^p$, $\Delta_z(t) \in \mathbb{Z}^\ell$, $\Delta_u(t) \in \mathbb{Z}^m$, and $\Delta_0 \in \mathbb{Z}^\ell$ satisfy (27). □

We defer the discussion about the error effect in the closed-loop system of the plant and the encrypted controller to the next subsection. It will be seen that if the closed-loop system is stable with respect to "perturbations", then the effect of the errors can be made arbitrarily small by an appropriate choice of the parameter L.

In the remaining, we discuss the strengths and weaknesses of the GSW-LWE encryption scheme. Compared to other LWE-based cryptosystems, an opportunity cost of using the GSW scheme is that it may require a large amount of storage; for example, if the parameters (q, σ_0, d) are chosen as $(q, \sigma_0, d) = (2^{48}, 10, 3)$ and the desired bit-security level is $\lambda = 80$, then the estimation (7) suggest that the dimension n should be chosen larger than $n = 7.16 \times 10^2$. It means that a scalar

Table 2
Computation times for GSW-LWE algorithms for a scalar variable, tested with MATLAB using an Intel i5 processor with 16 GB RAM.

(n, q, d)	$(50, 2^{48}, 3)$	$(1000, 2^{48}, 3)$
Enc/Dec	0.2 ms	2.6 ms
Enc$'$	15 ms	120 ms
GSW-LWE multiplication	0.7 ms	19.9 ms

message encrypted by GSW scheme turns into $(dn^2 \approx 1.54 \times 10^6)$-dimensional 48-bit numbers. Instead, besides the abilities of addition, multiplication, and recursive operation, another benefit of GSW-LWE scheme is that its implementation is simple and easy; in contrast to the cryptosystems requiring the modulus q chosen as a large odd number, there is no such restriction on the choice of the modulus q with the GSW-LWE scheme. Then, once choosing the modulus q and the base v chosen as powers of 2, it can be found that the encrypted addition and multiplication, and the algorithms Enc and Dec can be implemented with simple modular matrix multiplication, where the operations such as $(\cdot \mod q)$ or $D(\cdot)$ can be easily performed using binary bit operations.

An example of computation times consumed for each encrypted operation is shown in Table 2. In spite of a conservative choice of parameters to ensure the security, the operations Enc, Dec, and the multiplication turn out to be relatively fast. The encryption Enc$'$ takes a relatively long time, but considering that it will be used for encrypting the control parameters, it can be performed off-line, while the controller is initialized. The component-wise encryption and operation described in (21) would require a multiple of operation times as many as the size of the dimension, but a "packing method" (Genise, Gentry, Halevi, Li, & Micciancio, 2019) for the GSW-LWE scheme that encrypts a vector or a matrix into a "single ciphertext" can be used for reducing the computation time in practice.

3.2. Conversion of linear controllers to operate over $\mathbb{Z}_q$

The previous section showed that the dynamic operation of (24) can be implemented as (25) with all the parameters and the signals encrypted with the GSW-LWE scheme. Thus, the remaining task for the encrypted dynamic control implementation is to convert the "given" dynamic controller (4) over $\mathbb{R}$ (which is supposed to be designed in advance) to a system of the form (24) so that it can operate over the space $\mathbb{Z}_q$ based on the modular arithmetic.

Let the system (24) be rewritten as

$$z(t+1) = F \cdot z(t) + G \cdot Q(y(t)) \quad \mod q$$
$$u(t) = H \cdot z(t) + J \cdot Q(y(t)) \quad \mod q \tag{30}$$
$$z(0) = z_0 \in \mathbb{Z}_q^{\ell},$$

where $Q : \mathbb{R}^p \rightarrow \mathbb{Z}_q^p$ is a quantization function so that the input $y(t) \in \mathbb{R}^p$ of the given controller (4) can be regarded as the input of the system (30) as well. Now, given the controller (4), the problem of interest is to design the parameters $\{F, G, H, J, z_0\}$ of (30), so that the performance of the system (30) over $\mathbb{Z}_q$ is "practically equivalent" to that of the controller (4) over $\mathbb{R}$. The problem is more specifically stated, as follows.

Problem 1. Given the parameters $\{F, G, H, J, x_0\}$ of the system (4) consisting of real numbers together with a positive number $\epsilon > 0$, find parameters $\{F, G, H, J, z_0\}$ of integers and a function Q for the system (30), and find a function $g : \mathbb{Z}_q^m \rightarrow \mathbb{R}^m$, such that $\|u(t) - g(u(t))\| \leq \epsilon$ for all $t \geq 0$. □

It is desirable that, as in Fig. 2(b), only the encrypted signal of the plant output $y(t)$ is received at the controller, and only the encrypted output of $u(t)$ is transmitted back to the plant. Nonetheless, from the rationale that the output of the system is supposed to be decrypted at the actuator, an additional assumption can be considered that the

decrypted signal of $u(t)$ can be re-encrypted and transmitted back to the controller again; let the system (30) be replaced with the following form

$$z(t+1) = F \cdot z(t) + G \cdot Q(y(t)) + R \cdot Q'(u(t)) \quad \mod q$$
$$u(t) = H \cdot z(t) + J \cdot Q(y(t)) \quad \mod q \tag{31}$$
$$z(0) = z_0 \in \mathbb{Z}_q^{\ell}$$

where a (nonlinear) function $Q' : \mathbb{Z}_q^m \rightarrow \mathbb{Z}_q^m$ can be applied to the decrypted output before it is re-encrypted. See Fig. 12 describing the encrypted implementation of (31). Then, a relaxed problem assuming the "output re-encryption" can be reformulated as follows.

Problem 2. Given the parameters $\{F, G, H, J, x_0\}$ of (4) and $\epsilon > 0$, find a set of parameters $\{F, G, H, J, z_0, R\}$ of integers and functions $\{Q, Q'\}$ for the system (31) over $\mathbb{Z}_q$, and a function g, such that $\|g(u(t)) - u(t)\| \leq \epsilon$, $\forall t \geq 0$. □

Regarding the stated problems, we note that simply "scaling up" the parameters $\{F, G, H, J, x_0\}$ cannot be a solution. Consider the following example

$$
\begin{aligned}
x(t+1) &= -0.25 \times x(t) + 1 \\
u(t) &= x(t) \\
x(0) &= 1
\end{aligned}
\quad \rightarrow \quad
\begin{aligned}
u(1) &= 0.75, \\
u(2) &= 0.8125, \\
u(3) &= 0.796875, \\
u(4) &= 0.80078125.
\end{aligned}
\tag{32a}
$$

To keep the precision of the parameter 0.25 that is recursively multiplied with the state $x(t)$, let it be scaled by 100 and stored as an integer and the system is represented as

$$
\begin{aligned}
z(t+1) &= -25 \times z(t) + 100^{t+1} \\
u(t) &= z(t) \\
z(0) &= 1
\end{aligned}
\quad \rightarrow \quad
\begin{aligned}
u(1) &= 75, \\
u(2) &= 8125, \\
u(3) &= 796875, \\
u(4) &= 80078125,
\end{aligned}
\tag{32b}
$$

so that the output $u(t)$ is recovered as $u(t) = u(t)/100^t$. However, it can be seen that the size $|u(t)|$ of the output increases (exponentially) as time goes by, so that no matter how large the modulus q is chosen for the underlying space $\mathbb{Z}_q$ for the system (30), there will be an overflow and the output will be incorrect, in a finite time.

In general, a major issue related to the problem is *recursive multiplication of the state matrix consisting of non-integers*; the state matrix F of the given controller in (4), which is recursively multiplied with the state $x(t)$, should not be kept as $\lceil F/s \rceil \in \mathbb{Z}^{\ell \times \ell}$ with a scale factor $1/s > 1$ to keep the precision of the decimal part. If so, the state $z(t)$ of the implemented system will be multiplied by the scaled matrix $\lceil F/s \rceil$ together with the factor $1/s > 1$, so that the size of $z(t)$ (equivalently, the number of decimal places of the state $x(t)$) will increase as time goes by, even if the norm $\|x(t)\|$ of the state is bounded.

In this regard, from the following subsection, we first convert the state matrix to integers without use of scaling, and then convert the rest part of the system to integers. The methods to be introduced are presented in Kim, Shim, and Han (2022), which will be seen as a solution to Problem 2.

3.2.1. Conversion of state matrix to integers

A simple way of changing the parameters of a linear system while keeping the same input–output relation is the "similarity transformation". With an invertible matrix $T \in \mathbb{Z}^{\ell \times \ell}$, let the given controller (4) be transformed as

$$z(t+1) = TFT^{-1}z(t) + TGy(t)$$
$$u(t) = HT^{-1}z(t) + Jy(t) \tag{33}$$
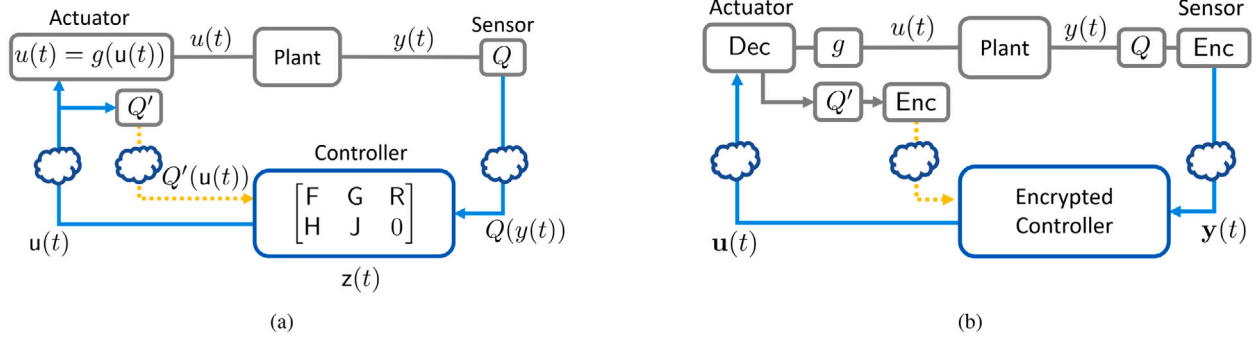$$z(0) = Tx_0$$

Fig. 12. (a) Block diagram of the system (30). (b) Encrypted implementation of (30) with output re-encryption.

which has the same input–output relation with the relation $z(t) = Tx(t)$. Then, to convert the given state matrix F to integers, one may try to find a transformation matrix T such that the state matrix $F \in \mathbb{R}^{\ell \times \ell}$ consisting of non-integers is turned into integers; i.e., $TFT^{-1} \in \mathbb{Z}^{\ell \times \ell}$.

However, this attempt will not be successful in general, due to the invariant characteristics of the linear systems. If there exists a transformation T such that $TFT^{-1} \in \mathbb{Z}^{\ell \times \ell}$, then it is obvious that the characteristic polynomial $\det(sI_\ell - TFT^{-1})$ of the matrix TFT^{-1} has the coefficients as integers. Thus, from the invariance that $\det(sI_\ell - TFT^{-1}) = \det(sI_\ell - F)$ no matter what the transform T is chosen, there does not exist a transformation T such that $TFT^{-1} \in \mathbb{Z}^{\ell \times \ell}$ as long as the given state matrix F is such that its characteristic polynomial $\det(sI_\ell - F)$ does not have integer coefficients.

Then, considering that the objective of Problem 2 is to convert the given controller into the form (31) where the output of the system can be treated as an "auxiliary input" at the same time, we first change the state matrix using the auxiliary input and then try to transform it to integers; let us observe that the system (4) can be rewritten as

$$x(t+1) = (F - RH)x(t) + (G - RJ)y(t) + Ru(t)$$
$$u(t) = Hx(t) + Jy(t) \tag{34}$$
$$x(0) = x_0$$

which has the same relation from the input $y(t)$ to the output $u(t)$, regardless of the choice of the matrix $R \in \mathbb{R}^{\ell \times m}$. Now, we consider the similarity transformation for the system (34), as

$$z(t+1) = T(F - RH)T^{-1}z(t) + T(G - RJ)y(t) + TRu(t) \tag{35a}$$
$$u(t) = HT^{-1}z(t) + Jy(t) \tag{35b}$$
$$z(0) = Tx_0 \tag{35c}$$

and try to find a transformation matrix T such that the "converted" state matrix consists of integers; that is,

$$T(F - RH)T^{-1} \in \mathbb{Z}^{\ell \times \ell}.$$

In what follows, it is proposed that the system of the form (34) can always be converted to have the state matrix as integers. First, without loss of generality, we can assume that the pair (F, H) of the given controller (4) is observable, i.e.,

$$\text{rank}\left(\begin{bmatrix} H \\ HF \\ \vdots \\ HF^{\ell-1} \end{bmatrix}\right) = \ell.$$

If it is not, "Kalman observable decomposition" can be considered so that the pair (F, H) can be reduced to an observable pair; an invertible matrix $W = [W_1^\top, W_2^\top]^\top \in \mathbb{R}^{\ell \times \ell}$ can always be found such that, with $z_1(t) = W_1 x(t) \in \mathbb{R}^{\ell'}$ and $z_2(t) = W_2 x(t) \in \mathbb{R}^{\ell - \ell'}$ with some $\ell' \leq \ell$, the controller (4) can be transformed into the form

$$z_1(t+1) = F_{11} z_1(t) + W_1 G y(t) \tag{36a}$$

$$z_2(t+1) = F_{21} z_1(t) + F_{22} z_2(t) + W_2 G y(t) \tag{36b}$$
$$u(t) = H_1 z_1(t) + J y(t) \tag{36c}$$

with

$$WFW^{-1} = \begin{bmatrix} F_{11} & 0_{\ell' \times (\ell - \ell')} \\ F_{21} & F_{22} \end{bmatrix} \quad \text{and} \quad HW^{-1} = \begin{bmatrix} H_1 & 0_{m \times (\ell - \ell')} \end{bmatrix},$$

where the reduced pair (F_{11}, H_1) becomes observable. Then, since the sub-state $z_2(t)$ of the unobservable part does not affect the value of the output $u(t)$ in (36c), it suffices to reduce the given controller (4) to the subsystem (36a) with (36c) only, which is observable and has the same input–output relation.

Now, based on the observability of the pair (F, H), it is proposed that, the matrices R and T can always be found such that the converted system (35) has the state matrix as integers; let the matrix R be found such that the eigenvalues $\{\lambda_i\}_{i=1}^{\ell}$ of the matrix $F - RH$ are distinct integers, for example. Then, by choosing the transformation matrix T that diagonalizes the matrix $F - RH$, we obtain

$$T(F - RH)T^{-1} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_\ell \end{bmatrix} \in \mathbb{Z}^{\ell \times \ell},$$

which is clearly a matrix consisting of integers. As a result, we have the following lemma.

Lemma 1. *For any observable pair $F \in \mathbb{R}^{\ell \times \ell}$ and $H \in \mathbb{R}^{m \times \ell}$, there exist matrices $R \in \mathbb{R}^{\ell \times m}$ and $T \in \mathbb{R}^{\ell \times \ell}$ such that $T(F - RH)T^{-1} \in \mathbb{Z}^{\ell \times \ell}$.* $\square$

If the controller (4) is an observable single-output system, the observable canonical form of the system can be simply used for the conversion. Indeed, let $H \in \mathbb{R}^{1 \times \ell}$, and let the transform matrix T be found such that the pair (F, H) is transformed into the form

$$TFT^{-1} = \begin{bmatrix} 0 & 0 & \cdots & 0 & a_1 \\ 1 & 0 & \cdots & 0 & a_2 \\ 0 & 1 & \cdots & 0 & a_3 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & a_\ell \end{bmatrix} \in \mathbb{R}^{\ell \times \ell},$$
$$HT^{-1} = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix} \in \mathbb{R}^{1 \times \ell}.$$

Then obviously, any matrix $R \in \mathbb{R}^{\ell \times 1}$, given by

$$R = T^{-1}\begin{bmatrix} a_1 - k_1 \\ a_2 - k_2 \\ \vdots \\ a_\ell - k_\ell \end{bmatrix}, \qquad k_i \in \mathbb{Z}, \quad i = 0, \ldots, \ell,$$

will yield the converted state matrix as integers, as

$$T(F - RH)T^{-1} = \begin{bmatrix} 0 & 0 & \cdots & 0 & k_1 \\ 1 & 0 & \cdots & 0 & k_2 \\ 0 & 1 & \cdots & 0 & k_3 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & k_\ell \end{bmatrix} \in \mathbb{Z}^{\ell \times \ell}.$$

3.2.2. Conversion to system over $\mathbb{Z}$

So far, we have converted the given controller (4) to the form (35), where the matrices T and R are found such that the state matrix $T(F - RH)T^{-1}$ consists of integers. Now, recalling that the overflow problem described in the example (32) was due to the state matrix F consisting of non-integers and due to the recursive multiplication of the state matrix $\lceil F/s \rceil \in \mathbb{Z}^{\ell \times \ell}$ scaled by $1/s > 1$, it can be expected that such problem would not be found because now the converted system (35) will keep the state matrix $T(F - RH)T^{-1}$ as integers without a scaling factor (that is, $1/s = 1$). In this subsection, we show how to implement the system having the state matrix as integers, to operate over the space $\mathbb{Z}$ using addition and multiplication.

The task is to convert all the signals and the parameters of the system (35) to integers using scale factors, except the state matrix which is already of integers. Note that, any matrix $G \in \mathbb{R}^{\ell \times p}$ (or any vector or scalar) can be stored as integers with arbitrary precision, as $\lceil G/s \rceil \in \mathbb{Z}^{\ell \times p}$, $1/s > 0$, because the error due to rounding can be arbitrary small as the scale factor $1/s$ tends to infinity (as s tends to zero), as

$$\left\| G - s \left\lceil \frac{G}{s} \right\rceil \right\| \le \frac{p}{2} \cdot s. \tag{37}$$

Thus, to keep the performance of the system (35) while scaling up the non-integer numbers therein, let $1/r > 0$, $1/s > 0$, and $L \in \mathbb{N}$ be scale factors, and let the factor $L/(rs)$ be multiplied to the both sides of the Eqs. (35a) and (35c), and the factor $L/(rs^2)$ be multiplied to the both sides of Eq. (35b), as

$$\frac{L \cdot z(t+1)}{rs} = T(F - RH)T^{-1} \cdot \frac{L \cdot z(t)}{rs} + \frac{T(G - RJ)}{s} \cdot \frac{L \cdot y(t)}{r}$$
$$+ \frac{TR}{s} \cdot \frac{L \cdot u(t)}{r}$$
$$\frac{L \cdot u(t)}{rs^2} = \frac{HT^{-1}}{s} \cdot \frac{L \cdot z(t)}{rs} + \frac{J}{s^2} \cdot \frac{L \cdot y(t)}{r} \tag{38}$$
$$\frac{L \cdot z(0)}{rs} = L \cdot \frac{Tx_0}{rs}.$$

Next, we take the rounding operation to the parameters and signals so that we consider the following system

$$\bar{z}(t+1) = \bar{F} \cdot \bar{z}(t) + \bar{G} \cdot \bar{y}(t) + \bar{R} \cdot L \cdot \left\lceil \frac{s^2}{L} \cdot \bar{u}(t) \right\rceil, \tag{39a}$$

$$\bar{u}(t) = \bar{H} \cdot \bar{z}(t) + \bar{J} \cdot \bar{y}(t) \tag{39b}$$

$$\bar{z}(0) = \bar{z}_0,$$

with

$$\bar{F} := T(F - RH)T^{-1}, \quad \bar{G} := \left\lceil \frac{T(G - RJ)}{s} \right\rceil, \quad \bar{R} := \left\lceil \frac{TR}{s} \right\rceil,$$
$$\bar{H} := \left\lceil \frac{HT^{-1}}{s} \right\rceil, \quad \bar{J} := \left\lceil \frac{J}{s^2} \right\rceil, \quad \bar{y}(t) := L \cdot \left\lceil \frac{y(t)}{r} \right\rceil, \quad \bar{z}_0 := L \cdot \left\lceil \frac{Tx_0}{rs} \right\rceil, \tag{40}$$

where the state $\bar{z}(t) \in \mathbb{Z}^\ell$, the input $\bar{y}(t) \in \mathbb{Z}^p$, and the output $\bar{u}(t) \in \mathbb{Z}^m$ will keep approximate values of the state $L \cdot z(t)/(rs)$, the input $L \cdot y(t)/r$, and the output $L \cdot u(t)/(rs^2)$ in (38), respectively. Note that, the constructed system (39) is defined over integers and operates using only addition and multiplication, except the operation $L \cdot \lceil (s^2/L) \cdot (\ ) \rceil$ which "divides" the value of $\bar{u}(t)$ by the factor $1/s^2$.

Next, we consider the performance of the system (39). Define $\tilde{z}(t) := (rs/L) \cdot \bar{z}(t)$ and $\tilde{u}(t) := (rs^2/L) \cdot \bar{u}(t)$. Then, the system (39) can be rewritten as

$$\tilde{z}(t+1) = T(F - RH)T^{-1}\tilde{z}(t) + T(G - RJ)y(t) + TR\tilde{u}(t) + e_z(t)$$
$$\tilde{u}(t) = HT^{-1}\tilde{z}(t) + Jy(t) + e_u(t) \tag{41}$$
$$\tilde{z}(0) = Tx_0 + e_{z,0}$$

where $e_z(t) \in \mathbb{R}^\ell$, $e_u(t) \in \mathbb{R}^m$, and $e_{z,0} \in \mathbb{R}^\ell$ denote the error due to the rounding (quantization), given by

$$e_z(t) = \left(T(G - RJ)y(t) - \left(s \left\lceil \frac{T(G - RJ)}{s} \right\rceil \right) \cdot \left(r \left\lceil \frac{y(t)}{r} \right\rceil \right) \right)$$
$$+ \left(TR\tilde{u}(t) - \left(s \left\lceil \frac{TR}{s} \right\rceil \right) \cdot \left(r \left\lceil \frac{\tilde{u}(t)}{r} \right\rceil \right) \right)$$
$$e_u(t) = \left(HT^{-1} - s \left\lceil \frac{HT^{-1}}{s} \right\rceil \right) \cdot \tilde{z}(t) + \left(Jy(t) - \left(s^2 \left\lceil \frac{J}{s^2} \right\rceil \right) \cdot \left(r \left\lceil \frac{y(t)}{r} \right\rceil \right) \right)$$
$$e_{z,0} = Tx_0 - rs \left\lceil \frac{Tx_0}{rs} \right\rceil. \tag{42}$$

We note that the factor L does not affect the values of $\{e_z(t), e_u(t), e_{z,0}\}$, so does nothing about the performance of (41); as an auxiliary scale factor, it will be used in Section 3.3, to deal with "error growth" of the LWE-based cryptosystem. One may suppose $L = 1$ in this subsection for simplicity.

Then, the performance of the implemented system (39) can be identified with the given controller model (4) with presence of perturbation; from (41), it can be easily verified that the state $\tilde{x}(t) := T^{-1}\tilde{z}(t)$ and the output $\tilde{u}(t)$ obey

$$\tilde{x}(t+1) = F\tilde{x}(t) + Gy(t) + e_x(t)$$
$$\tilde{u}(t) = H\tilde{x}(t) + Jy(t) + e_u(t) \tag{43}$$
$$\tilde{x}(0) = x_0 + e_0,$$

where the errors $e_x(t) \in \mathbb{R}^\ell$ and $e_0 \in \mathbb{R}^\ell$ are given by

$$e_x(t) = T^{-1}e_z(t) + TRe_u(t), \qquad e_0 = T^{-1}e_{z,0}. \tag{44}$$

Compared to the given controller (4) without considering the errors, note that $\tilde{x}(t) = x(t)$ and $\tilde{u}(t) = u(t)$, $\forall t \ge 0$, if $e_0 = 0$, $e_x(t) = 0$, and $e_u(t)$, $\forall t \ge 0$. The result is that the performance of (39) is equivalent to that of (43), with the relation

$$\tilde{x}(t) = \frac{rs}{L} \cdot T^{-1}\bar{z}(t), \qquad \tilde{u}(t) = \frac{rs^2}{L} \cdot \bar{u}(t). \tag{45}$$

In what follows, we discuss the effect of the perturbations $\{e_x(t), e_u(t), e_0\}$ on the performance of the systems (39) and (43). The claim is that, if the sizes of the errors $\{e_x(t), e_u(t), e_0\}$ can be made arbitrarily small by appropriate choice of the parameters $\{r, s\}$, and if the trajectories $\tilde{x}(t)$ and $\tilde{u}(t)$ of (43) are "stable" with respect to the perturbations $\{e_x(t), e_u(t), e_0\}$, then the difference of the trajectories $\tilde{x}(t)$ and $\tilde{u}(t)$ from those of the "ideal" trajectories $x(t)$ and $u(t)$ can be made arbitrarily small by the choice of $\{r, s\}$. And, this will ensure that the performance of the system (39) is "practically equivalent" to the given controller (4). In this regard, we make the following assumption.

Assumption 1. Given the controller (4), the perturbed model (43) satisfies the following: There exists a function $\delta(\epsilon) > 0$ such that for any $\epsilon > 0$, if

$$\|e_0\| \le \delta(\epsilon), \quad \|e_x(t)\| \le \delta(\epsilon), \quad \text{and} \quad \|e_u(t)\| \le \delta(\epsilon), \quad \forall t \ge 0, \tag{46}$$

then $\|\tilde{x}(t) - x(t)\| \le \epsilon$ and $\|\tilde{u}(t) - u(t)\| \le \epsilon$ hold for all $t \ge 0$. $\square$

Obviously, if the given controller (4) itself is stable (i.e., the state matrix F is Schur stable), then the condition of Assumption 1 is implied. But, we note that a system, in which the condition of Assumption 1 holds, may not be stable by itself (the matrix F may not be Schur stable). Typically, if the system (43) is a part of a closed-loop system, and if it is stable with respect to the perturbations $\{e_x(t), e_u(t), e_0\}$, then the condition of Assumption 1 will hold. In such cases, rigorously, the input $y(t)$ of the system may also be affected by the perturbations, but we omit such arguments for simplicity (See Kim et al. (2022, Section II.B)).

Finally, the following lemma states that the performance of the converted controller (39) operating over integers is guaranteed, provided

that the trajectories of the given controller is stable with respect to the quantization errors $\{e_x(t), e_u(t), e_0\}$. For a more rigorous proof under a more general setting, see Kim et al. (2022, Proposition 6).

Lemma 2. *For any $\epsilon > 0$, under Assumption 1, there exist $r' > 0$ and $s' > 0$ such that for any $r < r'$, $s < s'$, and $L \in \mathbb{N}$, the controller (39) implemented over $\mathbb{Z}$ guarantees that $\|(rs/L) \cdot T^{-1}\bar{z}(t) - x(t)\| \le \epsilon$ and $\|(rs^2/L) \cdot \bar{u}(t) - u(t)\| \le \epsilon$ hold, $\forall t \ge 0$.* $\square$

Sketch of Proof. Recall the relation (45) between the system (39) and (43), with the perturbations $\{e_x(t), e_u(t), e_0\}$ determined by (42) and (44). Given $\epsilon > 0$ and $\delta(\epsilon)$, we show that r' and s' can be chosen such that $r < r'$ and $s < s'$ ensure that (46) holds. Observe that, for any signal $y(t) \in \mathbb{R}^p$ and $G \in \mathbb{R}^{\ell \times p}$, some constants θ_1 and θ_2 can be found such that

$$\left\| Gy(t) - \left(s\left\lceil\frac{G}{s}\right\rceil\right)\cdot\left(r\left\lceil\frac{y(t)}{r}\right\rceil\right)\right\| \le \frac{\|G\|}{2}\cdot r + \frac{p}{2}\cdot\|y(t)\|\cdot s + \frac{p}{4}\cdot rs$$
$$\le \theta_1 \cdot \max\{\|y(t)\|, \theta_2\}\cdot\max\{r,s,rs\}.$$

Then, from (42) and (44), it can be verified that

$$\max\{\|e_x(t)\|, \|e_u(t)\|, \|e_0\|\}$$
$$\le \theta_1' \cdot \max\{\|y(t)\|, \|\tilde{x}(t)\|, \|\tilde{u}(t)\|, \theta_2'\}\cdot\max\{r, s, rs, s^2, rs^2\}$$
$$=: \alpha(\|y(t)\|, \|\tilde{x}(t)\|, \|\tilde{u}(t)\|, r, s) \tag{47}$$

with some constants θ_1' and θ_2'. From the boundedness of the signals of (4), let $\max\{\|y(t)\|, \|x(t)\|, \|u(t)\|\} \le M$ with some $M > 0$. Then, since $r = 0$ and $s = 0$ implies $\alpha = 0$ in (47), we can choose $r' > 0$ and $s' > 0$ such that

$$\alpha(M, M+\epsilon, M+\epsilon, r', s') \le \delta(\epsilon).$$

As a consequence, by Assumption 1, it can be proved that any $r < r'$ and $s < s'$ ensure that

$$\|\tilde{x}(t) - x(t)\| \le \epsilon, \quad \|\tilde{u}(t) - u(t)\| \le \epsilon, \quad \|\tilde{x}(t)\| \le M+\epsilon, \quad \|\tilde{u}(t)\| \le M+\epsilon,$$

for all $t \ge 0$, using mathematical induction. It completes the proof. ∎

3.2.3. Conversion to system over $\mathbb{Z}_q$

All the parameters and the signals of the given controller (4) are now converted to integers, in the controller (39). Recalling the goal of Problem 2, we map the parameters and signals of (39) to the set $\mathbb{Z}_q = \{0, 1, \dots, q-1\}$ of messages for encryption, and further convert the system (39) to the form (31), so that it can be directly encrypted using the result of Proposition 3.

First of all, we note that, given any dynamic system of the form

$$\bar{x}(t+1) = \overline{F}'\bar{x}(t) + \overline{G}'\bar{y}(t),$$
$$\bar{u}'(t) = \overline{H}'\bar{x}(t) + \overline{J}'\bar{y}(t), \tag{48}$$
$$\bar{x}(0) = \bar{x}_0,$$

where all the signals and parameters are of integers, the conversion to a system over $\mathbb{Z}_q$ itself is simple; just by taking the modulo operation to the parameters, the signals, and the computation outcomes of (48), it can be converted as

$$x(t+1) = F'x(t) + G'y(t) \mod q,$$
$$u'(t) = H'x(t) + J'y(t) \mod q, \tag{49}$$
$$x(0) = x_0,$$

where

$$F' = \overline{F}' \mod q, \quad G' = \overline{G}' \mod q, \quad H' = \overline{H}' \mod q,$$
$$J' = \overline{J}' \mod q, \quad y(t) = \bar{y}(t) \mod q, \quad x_0 = \bar{x}_0 \mod q. \tag{50}$$

Since the modulo operation is compatible with addition and multiplication, the conversion (49) ensures that

$$u'(t) = \bar{u}'(t) \mod q, \quad \forall t \ge 0. \tag{51}$$

Conversely, to recover the output $\bar{u}'(t)$ of arbitrary integers from the "projected" output $u'(t)$ consisting of elements of $\mathbb{Z}_q$, the "size" q of the space $\mathbb{Z}_q$ should be chosen appropriately. A sufficient condition is given in the following proposition; the original output $\bar{u}'(t)$ can be recovered from $u'(t)$ using the biased modulo operation defined in (1), if the size of the space $\mathbb{Z}_q$ "covers" the range of the output $\bar{u}'(t)$.

Proposition 4. *Consider two systems (48) and (49) with (50). Suppose that the output $\bar{u}'(t) = \text{col}\{\bar{u}_i'(t)\}_{i=1}^{m'}$, $m' \in \mathbb{N}$, of (48) be bounded as*

$$\bar{u}_i^{\min} \le \bar{u}_i'(t) \le \bar{u}_i^{\max}, \qquad \forall t \ge 0, \quad \forall i = 1, \dots, m', \tag{52}$$

with some constants $\{\bar{u}_i^{\min}\}_{i=1}^{m'}$ and $\{\bar{u}_i^{\max}\}_{i=1}^{m'}$. If

$$q \ge \bar{u}_i^{\max} - \bar{u}_i^{\min} + 1, \qquad \forall i = 1, \dots, m', \tag{53}$$

then $\bar{u}'(t) = u'(t) \mod(q, \text{col}\{\bar{u}_i^{\min}\}_{i=1}^{m'})$ holds for all $t \ge 0$. $\square$

Proof. Let the ith component of $u'(t) \mod(q, \text{col}\{\bar{u}_i^{\min}\}_{i=1}^{m'})$ be denoted by $\bar{u}_i''(t)$. Note that the operations $\mod q$ and $\mod(q, \text{col}\{\bar{u}_i^{\min}\}_{i=1}^{m'})$ add some multiples of q to the components only. Then, from (51), it is clear that $\bar{u}_i'(t) = \bar{u}_i''(t) + kq$ with some $k \in \mathbb{Z}$. Now, the definition (1) ensures that $\bar{u}_i^{\min} \le \bar{u}_i''(t) < \bar{u}_i^{\min} + q$, and (52) and (53) ensure that $\bar{u}_i^{\min} \le \bar{u}_i'(t) < \bar{u}_i^{\min} + q$. Hence, $k = 0$ and $\bar{u}_i''(t) = \bar{u}_i'(t)$. It completes the proof. ∎

Based on this observation, we convert the system (39) to the form (31) over $\mathbb{Z}_q$, and determine the parameters and functions therein, to find a solution to Problem 2. To determine the modulus q first, we calculate the range of the output $\bar{u}(t)$ of (39); from the boundedness of the output $u(t) = \text{col}\{u_i(t)\}_{i=1}^m$ of (4), let some constants $\{u_i^{\min}\}_{i=1}^m$ and $\{u_i^{\max}\}_{i=1}^m$ be found such that

$$u_i^{\min} \le u_i(t) \le u_i^{\max}, \qquad \forall i = 1, \dots, m, \quad \forall t \ge 0. \tag{54}$$

Then, given $\epsilon > 0$ from Problem 2, the design of (39) is supposed to ensure that the output $\bar{u}(t) = \text{col}\{\bar{u}_i(t)\}_{i=1}^m$ is bounded as $\|(rs^2/L)\cdot\bar{u}(t) - u(t)\| \le \epsilon$, which implies

$$L \cdot \frac{u_i^{\min} - \epsilon}{rs^2} \le \bar{u}_i(t) \le L \cdot \frac{u_i^{\max} + \epsilon}{rs^2}, \quad \forall i = 1, \dots, m. \tag{55}$$

Now, considering the condition (53), we choose the modulus q such that

$$q \ge \max_{i=1,\dots,m}\left\{L \cdot \frac{u_i^{\max} - u_i^{\min} + 2\epsilon}{rs^2}\right\} + 1. \tag{56}$$

Then, even if the operation $\mod q$ is taken to the output $\bar{u}(t)$, as $u'(t) = \bar{u}(t) \mod q$, the value of $\bar{u}(t)$ can be recovered from $u'(t)$, as

$$\bar{u}(t) = u'(t) \mod\left(q, \text{col}\left\{L \cdot \frac{u_i^{\min} - \epsilon}{rs^2}\right\}_{i=1}^m\right),$$

and a real-valued signal approximate to the real output $u(t) \approx (rs^2/L)\cdot\bar{u}(t)$ can also be computed from $u'(t)$, with a function $g : \mathbb{Z}_q^m \to \mathbb{R}^m$ defined as

$$g(u'(t)) := \frac{rs^2}{L}\cdot\left(u'(t) \mod\left(q, \text{col}\left\{L \cdot \frac{u_i^{\min} - \epsilon}{rs^2}\right\}_{i=1}^m\right)\right).$$

Finally, we convert the system (39) to a system over $\mathbb{Z}_q$, by simply taking the modulo operation, as

$$z(t+1) = F \cdot z(t) + G \cdot y(t) + R \cdot Q'(u(t)) \mod q \tag{57a}$$
$$u(t) = H \cdot z(t) + J \cdot y(t) \mod q \tag{57b}$$
$$z(0) = z_0$$

with

$$F = \overline{F} \mod q, \quad G = \overline{G} \mod q, \quad R = \overline{R} \mod q, \quad y(t) = \bar{y}(t) \mod q,$$
$$H = \overline{H} \mod q, \quad J = \overline{J} \mod q, \quad z_0 = \bar{z}_0 \mod q,$$

(58)

where the term $L \cdot \lceil (s^2/L) \cdot \bar{u}(t) \rfloor$ in (39) is replaced with the term $Q'(u(t))$, with a function $Q' : \mathbb{Z}_q^m \to \mathbb{Z}_q^m$ defined as

$$Q'(u(t))$$
$$:= L \cdot \left\lceil \frac{s^2}{L} \cdot \left(u(t) \mod \left(q, \text{col} \left\{ L \cdot \frac{u_i^{\min} - \epsilon}{rs^2} \right\}_{i=1}^m \right) \right) \right\rfloor \mod q. \quad (59)$$

Note that, since the operation $L \cdot \lceil (s^2/L) \cdot \bar{u}(t) \rfloor$ for $\bar{u}(t)$ is not compatible with the modulo operation, in (59), the value of $\bar{u}(t)$ is first computed from $u(t)$, and then the operation is applied. And, note that, with

$$y(t) = Q(y(t)) := L \cdot \left\lceil \frac{y(t)}{r} \right\rfloor \mod q,$$

the constructed system (57) takes the same form that Problem 2 considers.

As the end result of Section 3.2, and as a solution to Problem 2, the following theorem shows that the performance of the implemented system (57), which operates based on modular arithmetic over the space $\mathbb{Z}_q$, can be arbitrarily close to that of the given system (4), by increasing the scale factors $1/r$ and $1/s$, regardless of the factor L.

Theorem 1. *Given any controller (4) and $\epsilon > 0$, under Assumption 1, there exist $r' > 0$ and $s' > 0$ such that for any $r < r'$, $s < s'$, and $L \in \mathbb{N}$, the controller (57) implemented over $\mathbb{Z}_q$ with (56) guarantees $\|g(u(t)) - u(t)\| \leq \epsilon, \forall t \geq 0$.* □

Proof. Consider the system (39) over $\mathbb{Z}$ as an auxiliary system. Thanks to Lemma 2, we choose r' and s' such that any $r < r'$ and $s < s'$ ensure that $\|(rs^2/L) \cdot \bar{u}(t) - u(t)\| \leq \epsilon, \forall t \geq 0$. By the boundedness of the output $u(t)$ as (54) with the constants $\{u_i^{\min}, u_i^{\max}\}_{i=1}^m$, it follows that (55) holds. Now, it is enough to show that

$$\bar{u}(t) = \left(u(t) \mod \left(q, \text{col} \left\{ L \cdot \frac{u_i^{\min} - \epsilon}{rs^2} \right\}_{i=1}^m \right) \right) \quad (60)$$

for all $t \geq 0$. Note that $y(t) = \bar{y}(t) \mod q, \forall t \geq 0$, and $z_0 = \bar{z}_0 \mod q$, so, from (39b) and (57b), it follows that $u(0) = \bar{u}(0) \mod q$. Now, suppose that $u(\tau) = \bar{u}(\tau) \mod q$ for some $\tau \geq 0$. Then, the condition (56) ensures that (60) holds for $t = \tau$, which can be proved analogously to the proof of Proposition 4. This is followed by $Q'(u(\tau)) = L \cdot \lceil (s^2/L) \cdot (\bar{u}(\tau)) \rfloor$, so that it can be easily verified that $z(\tau+1) = \bar{z}(\tau+1) \mod q$, by comparing (39a) and (57a). And, from (39b) and (57b), it is followed by $u(\tau+1) = \bar{u}(\tau+1) \mod q$. Hence, the proof is completed. ∎

The result of Theorem 1 can be summarized as follows. First of all, from the example in (32), it has been observed that linear systems having the state matrix as non-integers cannot continue the encrypted operation for an infinite time horizon, because of the limitation of homomorphic encryption. Then, based on the assumption that the encrypted controller can receive a re-encrypted signal of its output, Lemma 1 has converted the state matrix to integers in (35), while keeping the same relation from the input $y(t)$ to the output $u(t)$. Next, the quantization parameters r and s have been introduced for the rest controller matrices and signals. And, Lemma 2 and Theorem 1 showed that linear systems having the state matrix as integers can be converted to operate over $\mathbb{Z}_q$, based on modular addition and multiplication, so that they can be implemented over encrypted data. The performance is guaranteed and its error can be made arbitrarily small, as long as the modulus q for the space $\mathbb{Z}_q$ is chosen as (56), so that it covers the range of the output.

A follow-up result proposing a solution to Problem 1 can be found in Kim, Shim, Sandberg, and Johansson (2021), where the parameters $\{F, G, H, J\}$ are found as time-varying matrices consisting of integers.

3.3. Encrypted dynamic system and controlled error growth

Finally, we combine the GSW-LWE cryptosystem (described in Section 3.1) to the implemented controller (57) over $\mathbb{Z}_q$ (constructed in Section 3.2), and discuss the result.

Thanks to Proposition 3 and Theorem 1, the result is straightforward; let the matrices of the controller (57) be encrypted using the GSW encryption $\text{Enc}' : \mathbb{Z}_q \to \mathbb{Z}_q^{n \times dn}$ component-wisely, as

$$\mathbf{F} := \text{Enc}'(F), \quad \mathbf{G} := \text{Enc}'(G), \quad \mathbf{R} := \text{Enc}'(R), \quad \mathbf{H} := \text{Enc}'(H), \quad \mathbf{J} := \text{Enc}'(J),$$

and let the initial value $z_0 \in \mathbb{Z}_q^\ell$ of the controller state and the controller input $y(t) \in \mathbb{R}^p$ be quantized using the function $Q : \mathbb{R}^p \to \mathbb{Z}_q^p$ and encrypted using the LWE-based encryption $\text{Enc} : \mathbb{Z}_q \to \mathbb{Z}_q^n$, component-wisely, as

$$\mathbf{z}_0 := \text{Enc}(z_0), \qquad \mathbf{y}(t) := \text{Enc}(Q(y(t))).$$

Then, the encrypted controller is constructed as

$$\mathbf{z}(t+1) = \mathbf{F} \cdot D(\mathbf{z}(t)) + \mathbf{G} \cdot D(\mathbf{y}(t)) + \mathbf{R} \cdot D(\mathbf{u}'(t)) \mod q$$
$$\mathbf{u}(t) = \mathbf{H} \cdot D(\mathbf{z}(t)) + \mathbf{J} \cdot D(\mathbf{y}(t)) \mod q \quad (61)$$
$$\mathbf{z}(0) = \mathbf{z}_0$$

where $\mathbf{z}(t) \in \mathbb{Z}_q^{\ell n}$ and $\mathbf{u}(t) \in \mathbb{Z}_q^{mn}$ are the encrypted state and output, respectively, and $\mathbf{u}'(t) \in \mathbb{Z}_q^{mn}$ is the re-encrypted signal of the output $\mathbf{u}(t)$ with the function $Q' : \mathbb{Z}_q^m \to \mathbb{Z}_q^m$, defined as

$$\mathbf{u}'(t) := \text{Enc}(Q'(\text{Dec}(\mathbf{u}(t)))). \quad (62)$$

Note that the operation of the encrypted controller (61) coincides with the description of Fig. 12(b).

From now on, we discuss the effect of the "injected errors" of the cryptosystem, which has been seen in Proposition 3. Recall that, in Corollary 1, the factor L has been used for scaling up the messages, in order to reduce the effect of the errors. For this purpose, in fact, all the signals of the systems (39) and (57) have already been "scaled up by L", although it did nothing about the performance in the previous subsection. Let us consider the performance of the controller (61); let the decryption Dec be taken to both sides of (61). Analogously to Proposition 3, the decrypted state $\tilde{z}(t) := \text{Dec}(\mathbf{z}(t)) \in \mathbb{Z}_q^\ell$ and the decrypted output $\tilde{u}(t) := \text{Dec}(\mathbf{u}(t))$ will obey

$$\tilde{z}(t+1) = F \cdot \tilde{z}(t) + G \cdot (Q(y(t)) + \Delta_y(t))$$
$$+ R \cdot (Q'(\tilde{u}(t)) + \Delta_{u'}(t)) + \Delta_z(t) \mod q$$
$$\tilde{u}(t) = H \cdot \tilde{z}(t) + J(Q(y(t)) + \Delta_y(t)) + \Delta_u(t) \mod q \quad (63)$$
$$\tilde{z}(0) = z_0 + \Delta_0 \mod q$$

with some $\Delta_y(t) \in \mathbb{Z}^p$, $\Delta_{u'}(t) \in \mathbb{Z}^m$, $\Delta_z(t) \in \mathbb{Z}^\ell$, $\Delta_u(t) \in \mathbb{Z}^m$, and $\Delta_0 \in \mathbb{Z}^m$ such that

$$\|\Delta_y(t)\| \leq n_0 \sigma, \qquad \|\Delta_{u'}(t)\| \leq n_0 \sigma, \qquad \|\Delta_z(t)\| \leq (\ell + p)\Delta_{\text{Mult}},$$
$$\|\Delta_u(t)\| \leq (\ell + p)\Delta_{\text{Mult}}, \qquad \|\Delta_0\| \leq n_0 \sigma.$$

To make use of the results of Theorem 1 and Lemma 2, we consider an auxiliary system defined over $\mathbb{Z}$, as

$$\bar{z}(t+1) = \overline{F} \cdot \bar{z}(t) + \overline{G} \cdot (\bar{y}(t) + \Delta_y(t))$$
$$+ \overline{R} \cdot \left(L \cdot \left\lceil \frac{s^2}{L} \cdot \bar{u}(t) \right\rfloor + \Delta_{u'}(t) \right) + \Delta_z(t)$$
$$\bar{u}(t) = \overline{H} \cdot \bar{z}(t) + J(\bar{y}(t) + \Delta_y(t)) + \Delta_u(t) \quad (64)$$
$$\bar{z}(0) = \bar{z}_0 + \Delta_0.$$

Then, analogously to Lemma 2, it can be verified that the state $\tilde{x}(t)$ and the output $\tilde{u}(t)$ defined as (45) will obey

$$\tilde{x}(t+1) = F\tilde{x}(t) + Gy(t) + e'_x(t)$$
$$\tilde{u}(t) = H\tilde{x}(t) + Jy(t) + e'_u(t) \quad (65)$$
$$\tilde{x}(0) = e'_0$$

where the errors $\{e'_x(t), e'_u(t), e'_0\}$ are determined as

$$e'_x(t) = e_x(t) + \Delta^{\mathsf{L}}_x(t) := e_x(t) + \frac{\mathsf{rs} \cdot T^{-1} \cdot (\overline{G}\Delta_y(t) + \overline{R}\Delta_{u'}(t) + \Delta_z(t))}{\mathsf{L}}$$
$$+ \frac{\mathsf{rs}^2 \cdot T R(\overline{J}\Delta_y(t) + \Delta_u(t))}{\mathsf{L}}$$

$$e'_u(t) = e_u(t) + \Delta^{\mathsf{L}}_u(t) := e_u(t) + \frac{\mathsf{rs}^2 \cdot (\overline{J}\Delta_y(t) + \Delta_u(t))}{\mathsf{L}}$$

$$e'_0 = e_0 + \Delta^{\mathsf{L}}_0 := e_0 + \frac{\mathsf{rs} \cdot T^{-1} \cdot \Delta_0}{\mathsf{L}},$$

where $\{e_x(t), e_u(t), e_0\}$ are the same quantization errors defined in (42) and (44). Note that $\{\Delta^{\mathsf{L}}_x(t), \Delta^{\mathsf{L}}_u(t), \Delta^{\mathsf{L}}_0\}$ are bounded as

$$\max\{\|\Delta^{\mathsf{L}}_x(t)\|, \|\Delta^{\mathsf{L}}_u(t)\|, \|\Delta^{\mathsf{L}}_0\|\}$$
$$\leq \frac{\mathsf{r}}{\mathsf{L}} \cdot \max\{\|T^{-1}\|, \|T R\|, 1\}$$
$$\cdot ((\max\{\|G\| + \|R\|, \|J\|\} + 1)n_0\sigma + (\ell + \mathsf{p})\Delta_{\mathsf{Mult}})$$
$$=: \beta(\mathsf{r}, \mathsf{L}).$$

Now, in the following theorem, we prove that the performance of the controller (61) is the same with that of (65), where the errors $\{e'_x(t), e'_u(t), e'_0\}$ consist of the errors $\{e_x(t), e_u(t), e_0\}$ due to quantization, and the rest due to error injection of the cryptosystem. Since only the signals are scaled by $\mathsf{L} \in \mathbb{N}$ before encryption and the injected errors are not, it can be understood that the effect of the cryptosystem errors can be made arbitrarily small, by increasing the factor L.

Theorem 2. *Given any controller* (4) *and* $\epsilon > 0$, *under* Assumption 1, *there exist* $\mathsf{r}' > 0$, $\mathsf{s}' > 0$, *and* $\mathsf{L}' \in \mathbb{N}$ *such that for any* $\mathsf{r} < \mathsf{r}'$, $\mathsf{s} < \mathsf{s}'$, *and* $\mathsf{L} > \mathsf{L}'$, *the encrypted controller* (61) *with* (56) *ensures* $\|g(\mathsf{Dec}(\mathbf{u}(t))) - u(t)\| \leq \epsilon$, $\forall t \geq 0$. $\square$

Sketch of Proof. The proof is analogous to the proof of Theorem 1. Given $\delta(\epsilon) > 0$ from Assumption 1, analogously to Lemma 2 with (47), we can choose r', s', and L' such that

$$\alpha(M, M + \epsilon, M + \epsilon, \mathsf{r}', \mathsf{s}') + \beta(\mathsf{r}', \mathsf{L}') \leq \delta(\epsilon).$$

This ensures that the auxiliary system (64) satisfies $\|(\mathsf{rs}^2/\mathsf{L}) \cdot \overline{u}(t) - u(t)\| \leq \epsilon$, provided that Assumption 1 holds. Then, it follows that (55) holds, so that analogously to the proofs of Proposition 4 and Theorem 1, the condition (56) guarantees that (60) holds for all $t \geq 0$. It follows that

$$g(\mathsf{Dec}(\mathbf{u}(t))) = g(\tilde{\mathsf{z}}(t)) = \frac{\mathsf{rs}^2}{\mathsf{L}} \cdot \overline{u}(t)$$

holds for all $t \geq 0$, so the proof is completed. $\blacksquare$

Finally, we review the implications of Theorem 2. There have been two issues that hinder the unlimited operation of linear dynamic systems implemented over encrypted data; the first issue was due to the recursive multiplication by non-integer numbers, and it has been handled by the introduced method of converting the state matrix to integers.

Then, the second issue was the "error growth" problem of the cryptosystems. Even though the given state matrix of the system consists of only integers, most homomorphic cryptosystems (that allow for both the addition and multiplication) do not support the recursive multiplication by the encrypted numbers for an infinite number of times, unless bootstrapping techniques of fully homomorphic encryption is utilized. And, this is because of the error growth problem; once the injected errors exceed a certain bound, then the correct computation outcome cannot be expected.

To allow for the unlimited recursive multiplication, the benefit of GSW-LWE cryptosystem has been first discussed. Considering that linear systems only update the state (multiplicand) and does not update the matrix (multiplier), it has been observed that the matrix encrypted using GSW scheme can be multiplied to the LWE-based encrypted

vectors for an infinite number of times, thanks to the structure of LWE-type ciphertexts where the errors are stored together with the messages. And, despite the recursive multiplication repeated unlimited times, it has been shown that the growth of the injected errors does not tend to be infinitely large, and its effect is suppressed and controlled, under stability. The effect of the errors can be identified with perturbations (or external disturbances), and its size can be made small by appropriate choice of parameters, so that it can keep the performance as practically equivalent to the given un-encrypted model.

Compared to the methods from cryptography utilizing the bootstrapping techniques of fully homomorphic encryptions, the introduced method exploits addition and multiplication over ciphertexts only, which does not require a substantial amount of computational resources. A couple of related results are introduced; a guideline of choosing the cryptosystem parameters (n, q, σ_0) and the control parameters $(\mathsf{r}, \mathsf{s}, \mathsf{L})$ that guarantees both the desired level of security and control performance is found in Kim, Shim, and Han (2020b). And, further illustrative explanations using MATLAB example codes can be found in Kim, Shim, and Han (2020a).

4. Conclusion

We have introduced several approaches for encrypted control, based on homomorphic encryption, multi-party computation, and secret sharing. We have compared their benefits and weaknesses, and discussed the trade-offs between them. It has been suggested that the encrypted control approach and the corresponding cryptosystem should be chosen with an engineering point of view, so that its security model, enabled operations, and computation efficiency is suitable for the system under construction.

Among the research directions struggling to overcome the trade-offs between security, computation efficiency, and the operation ability, we have introduced a homomorphic encryption based problem which aims for implementing linear dynamic systems exploiting only addition and multiplication over integers. Then, it has been addressed that the problems from cryptography can be tackled with the control perspectives; we have addressed that the error growth of the GSW-LWE cryptosystem can be regarded as perturbations in control systems, and the issue of incapability of recursive multiplication by non-integer state matrix has been resolved using a method based on pole-placement technique.

References

Alexandru, A. B., Morari, M., & Pappas, G. J. (2018). Cloud-based MPC with encrypted data. In *2018 IEEE conference on decision and control (CDC)* (pp. 5014–5019). IEEE.

Alexandru, A. B., & Pappas, G. J. (2020). Secure multi-party computation for cloud-based control. In *Privacy in dynamical systems* (pp. 179–207). Springer.

Alexandru, A. B., Tsiamis, A., & Pappas, G. J. (2020). Towards private data-driven control. In *2020 59th IEEE conference on decision and control (CDC)* (pp. 5449–5456). IEEE.

Amin, S., Cárdenas, A. A., & Sastry, S. S. (2009). Safe and secure networked control systems under denial-of-service attacks. In *International workshop on hybrid systems: computation and control* (pp. 31–45). Springer.

Beimel, A. (2011). Secret-sharing schemes: A survey. In *International conference on coding and cryptology* (pp. 11–46). Springer.

Bogetoft, P., Christensen, D. L., Damgård, I., Geisler, M., Jakobsen, T., Krøigaard, M., et al. (2009). Secure multiparty computation goes live. In *International conference on financial cryptography and data security* (pp. 325–343). Springer.

Boneh, D., Goh, E.-J., & Nissim, K. (2005). Evaluating 2-DNF formulas on ciphertexts. In *Theory of cryptography conference* (pp. 325–341). Springer.

Bourbakis, N., & Alexopoulos, C. (1992). Picture data encryption using scan patterns. *Pattern Recognition, 25*(6), 567–581.

Brakerski, Z., Gentry, C., & Vaikuntanathan, V. (2014). (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT), 6*(3), 1–36.

Brakerski, Z., & Vaikuntanathan, V. (2011). Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *Annual cryptology conference* (pp. 505–524). Springer.

Brickell, E. F. (1989). Some ideal secret sharing schemes. In *Workshop on the theory and application of of cryptographic techniques* (pp. 468–475). Springer.

Chen, L., Chen, L., Jordan, S., Liu, Y.-K., Moody, D., Peralta, R., et al. (2016). *Report on post-quantum cryptography, Vol. 12*. US Department of Commerce, National Institute of Standards and Technology.

Cheon, J. H., Kim, A., Kim, M., & Song, Y. (2017). Homomorphic encryption for arithmetic of approximate numbers. In *International conference on the theory and application of cryptology and information security* (pp. 409–437). Springer.

Chillotti, I., Gama, N., Georgieva, M., & Izabachene, M. (2016). Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *International conference on the theory and application of cryptology and information security* (pp. 3–33). Springer.

Cramer, R., Damgård, I. B., et al. (2015). *Secure multiparty computation*. Cambridge University Press.

Dahl, M., Ning, C., & Toft, T. (2012). On secure two-party integer division. In *International conference on financial cryptography and data security* (pp. 164–178). Springer.

Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J. B., & Toft, T. (2006). Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In *Theory of cryptography conference* (pp. 285–304). Springer.

Dijk, M. v., Gentry, C., Halevi, S., & Vaikuntanathan, V. (2010). Fully homomorphic encryption over the integers. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 24–43). Springer.

Ducas, L., & Micciancio, D. (2015). FHEW: bootstrapping homomorphic encryption in less than a second. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 617–640). Springer.

Dwork, C. (2008). Differential privacy: A survey of results. In *International conference on theory and applications of models of computation* (pp. 1–19). Springer.

Dwork, C., Roth, A., et al. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science, 9*(3–4), 211–407.

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory, 31*(4), 469–472.

Fan, J., & Vercauteren, F. (2012). Somewhat practical fully homomorphic encryption. *Cryptology EPrint Archive*.

Farokhi, F., Shames, I., & Batterham, N. (2017). Secure and private control using semi-homomorphic encryption. *Control Engineering Practice, 67*, 13–20.

Fritz, R., Fauser, M., & Zhang, P. (2019). Controller encryption for discrete event systems. In *2019 American control conference (ACC)* (pp. 5633–5638). IEEE.

Genise, N., Gentry, C., Halevi, S., Li, B., & Micciancio, D. (2019). Homomorphic encryption for finite automata. In *International conference on the theory and application of cryptology and information security* (pp. 473–502). Springer.

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on theory of computing* (pp. 169–178).

Gentry, C., Halevi, S., & Smart, N. P. (2012). Better bootstrapping in fully homomorphic encryption. In *International workshop on public key cryptography* (pp. 1–16). Springer.

Gentry, C., Sahai, A., & Waters, B. (2013). Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Annual cryptology conference* (pp. 75–92). Springer.

Gonzalez-Serrano, F. J., Amor-Martın, A., & Casamayon-Anton, J. (2014). State estimation using an extended Kalman filter with privacy-protected observed inputs. In *2014 IEEE international workshop on information forensics and security (WIFS)* (pp. 54–59). IEEE.

Karnin, E., Greene, J., & Hellman, M. (1983). On secret sharing systems. *IEEE Transactions on Information Theory, 29*(1), 35–41.

Kim, J., Lee, C., Shim, H., Cheon, J. H., Kim, A., Kim, M., et al. (2016). Encrypting controller using fully homomorphic encryption for security of cyber-physical systems. *IFAC-PapersOnLine, 49*(22), 175–180.

Kim, J., Shim, H., & Han, K. (2020a). Comprehensive introduction to fully homomorphic encryption for dynamic feedback controller via LWE-based cryptosystem. In *Privacy in dynamical systems* (pp. 209–230). Springer.

Kim, J., Shim, H., & Han, K. (2020b). Design procedure for dynamic controllers based on LWE-based homomorphic encryption to operate for infinite time horizon. In *2020 59th IEEE conference on decision and control (CDC)* (pp. 5463–5468). IEEE.

Kim, J., Shim, H., & Han, K. (2022). Dynamic controller that operates over homomorphically encrypted data for infinite time horizon. *IEEE Transactions on Automatic Control*.

Kim, J., Shim, H., Sandberg, H., & Johansson, K. H. (2021). Method for running dynamic systems over encrypted data for infinite time horizon without bootstrapping and re-encryption. In *2021 60th IEEE conference on decision and control (CDC)* (pp. 5614–5619). IEEE.

Kogiso, K., & Fujita, T. (2015). Cyber-security enhancement of networked control systems using homomorphic encryption. In *2015 54th IEEE conference on decision and control (CDC)* (pp. 6836–6843). IEEE.

Langner, R. (2011). Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy, 9*(3), 49–51.

Lindell, Y. (2017). How to simulate it–a tutorial on the simulation proof technique. *Tutorials on the Foundations of Cryptography*, 277–346.

Lindner, R., & Peikert, C. (2011). Better key sizes (and attacks) for LWE-based encryption. In *Cryptographers' track at the RSA conference* (pp. 319–339). Springer.

Liu, Y., Ning, P., & Reiter, M. K. (2011). False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security, 14*(1), 1–33.

Lyubashevsky, V., Peikert, C., & Regev, O. (2010). On ideal lattices and learning with errors over rings. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 1–23). Springer.

Matthews, R. (1989). On the derivation of a "chaotic" encryption algorithm. *Cryptologia, 13*(1), 29–42.

Mo, Y., & Murray, R. M. (2016). Privacy preserving average consensus. *IEEE Transactions on Automatic Control, 62*(2), 753–765.

Mo, Y., & Sinopoli, B. (2009). Secure control against replay attacks. In *2009 47th annual allerton conference on communication, control, and computing (Allerton)* (pp. 911–918). IEEE.

Nishide, T., & Ohta, K. (2007). Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In *International workshop on public key cryptography* (pp. 343–360). Springer.

Nozari, E., Tallapragada, P., & Cortés, J. (2017). Differentially private average consensus: Obstructions, trade-offs, and optimal algorithm design. *Automatica, 81*, 221–231.

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques* (pp. 223–238). Springer.

Pasqualetti, F., Dörfler, F., & Bullo, F. (2013). Attack detection and identification in cyber-physical systems. *IEEE Transactions on Automatic Control, 58*(11), 2715–2729.

Regev, O. (2009). On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM, 56*(6), 1–40.

Rivest, R. L., Adleman, L., Dertouzos, M. L., et al. (1978). On data banks and privacy homomorphisms. *Foundations of Secure Computation, 4*(11), 169–180.

Sandberg, H., Amin, S., & Johansson, K. H. (2015). Cyberphysical security in networked control systems: An introduction to the issue. *IEEE Control Systems Magazine, 35*(1), 20–23.

Schlor, S., Hertneck, M., Wildhagen, S., & Allgöwer, F. (2021). Multi-party computation enables secure polynomial control based solely on secret-sharing. In *2021 60th IEEE conference on decision and control (CDC)* (pp. 4882–4887). IEEE.

Schulze Darup, M., Alexandru, A. B., Quevedo, D. E., & Pappas, G. J. (2021). Encrypted control for networked systems: An illustrative introduction and current challenges. *IEEE Control Systems Magazine, 41*(3), 58–78.

Schulze Darup, M., & Jager, T. (2019). Encrypted cloud-based control using secret sharing with one-time pads. In *2019 IEEE 58th conference on decision and control (CDC)* (pp. 7215–7221). IEEE.

Schulze Darup, M., Redder, A., Shames, I., Farokhi, F., & Quevedo, D. (2017). Towards encrypted MPC for linear constrained systems. *IEEE Control Systems Letters, 2*(2), 195–200.

Shamir, A. (1979). How to share a secret. *Communications of the ACM, 22*(11), 612–613.

Slay, J., & Miller, M. (2007). Lessons learned from the maroochy water breach. In *International conference on critical infrastructure protection* (pp. 73–82). Springer.

Smith, R. S. (2015). Covert misappropriation of networked control systems: Presenting a feedback structure. *IEEE Control Systems Magazine, 35*(1), 82–92.

Suh, J., & Tanaka, T. (2021). Encrypted value iteration and temporal difference learning over leveled homomorphic encryption. In *2021 American control conference (ACC)* (pp. 2555–2561). IEEE.

Teixeira, A., Shames, I., Sandberg, H., & Johansson, K. H. (2015). A secure control framework for resource-limited adversaries. *Automatica, 51*, 135–148.

Wang, Y. (2019). Privacy-preserving average consensus via state decomposition. *IEEE Transactions on Automatic Control, 64*(11), 4711–4716.

Zhou, S., Yu, Z., Nasr, E. S. A., Mahmoud, H. A., Awwad, E. M., & Wu, N. (2020). Homomorphic encryption of supervisory control systems using automata. *IEEE Access, 8*, 147185–147198.